Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

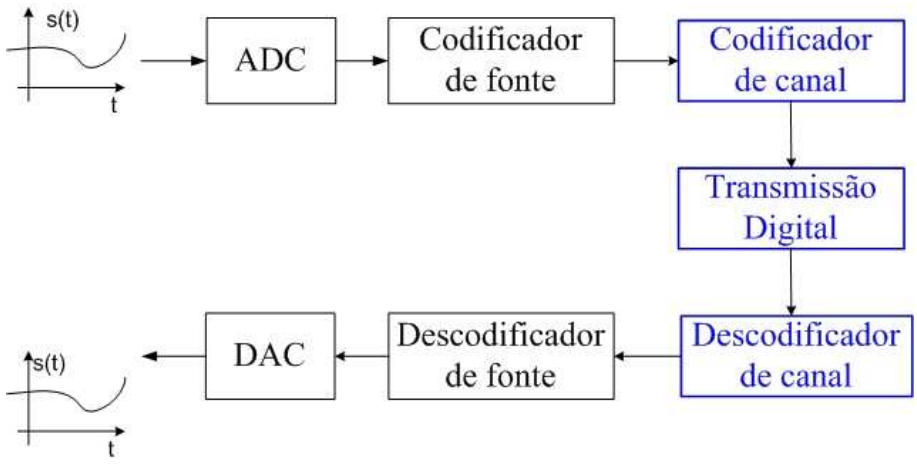
Licenciatura em Engenharia Informática e de Computadores

Comunicações

Códigos detectores e correctores de erros

Artur Ferreira

11 de Dezembro de 2007



Nota de agradecimento
O autor agradece ao prof. André Lourenço pelas sugestões, críticas e comentários que ajudaram a melhorar a qualidade deste texto.

Índice

Lista de Figuras

Lista de Tabelas

1 Introdução

Este documento apresenta o conceito de codificação de canal, seguindo a abordagem adoptada na unidade curricular Comunicações, do 3º semestre da LEIC. Os tópicos abordados são o conceito de codificação de canal e os códigos detectores e correctores de erros, bem como as suas aplicações. Apresentam-se os principais tópicos abordados da referida unidade curricular. Contudo, este documento não dispensa a consulta da bibliografia recomendada [1, 10], com especial ênfase nos capítulos 1, 2 e 3 de [10]. Pretende-se assim que este documento em conjunto com o livro [11] contenham todos os assuntos leccionados na referida unidade curricular.

Caracterizam-se canais de comunicação através de modelos probabilísticos. Introduzem-se os códigos detectores e correctores de erros e as motivações para a sua existência. Descrevem-se os códigos, bem como os processos de codificação e descodificação e apresentam-se exemplos. Tratam-se códigos lineares de bloco sistemáticos e não sistemáticos, referindo aplicações dos mesmos. Analisam-se também os códigos cíclicos, enquanto sub-classe dos códigos lineares de bloco, dando ênfase às suas capacidades de detecção de erros e aplicações.

Na secção 2 introduz-se o conceito de codificação de canal, caracterizando o canal através de modelo probabilístico, enquadrando-o no diagrama de blocos geral do processo de comunicação. A secção 3 aparesenta as características elementares dos códigos de bloco e dos seus processos de codificação e descodificação. A secção 4 trata os códigos lineares de bloco, as suas características e realização na forma matricial dos processos de codificação e descodificação. Aborda-se ainda a técnica de modificação de código, a qual consiste em realizar alterações pontuais a códigos estabelecidos. Na secção 5 apresenta-se os códigos lineares de bloco cíclicos, enquanto sub-classe dos códigos lineares de bloco. Analisam-se as palavras de código como polinómios. Definem-se códigos através do respectivo polinómio gerador. Apresentam-se códigos cíclicos na forma sistemática e não sistemática. Exemplifica-se a realização de codificadores e descodificadores em *hardware*. A secção 6 apresenta a utilização do MATLAB para operações ilustrativas dos conceitos abordados ao longo do texto. A secção 7 elenca aplicações dos códigos. Finalmente, o anexo A refere-se o cálculo do dígito de controlo do BI (Bilhete de Identidade) e do dígito de controlo do ISBN (International Standard Book Number), enquanto que o anexo B apresenta um conjunto de exercícios propostos.

As secções assinaladas com * referem-se a tópicos de menor importância, no contexto da unidade curricular Comunicações. Estes tópicos não são leccionados, com profundidade, na unidade curricular.

2 Codificação de canal

A figura 1 apresenta o cenário do processo de comunicação digital. Tendo em conta que se pretende transmitir o sinal analógico $s(t)$, através de um sistema de comunicação digital, é necessário converter este sinal do domínio analógico para o domínio digital, ou seja, o sinal analógico deve ser convertido numa sequência de bits. Esta acção designa-se por digitalização e é realizada através de um dispositivo designado por ADC-*Analog to Digital Converter* [11][1]. A sequência binária resultante da digitalização dá entrada no codificador de fonte, de forma a ser transformada noutra de menor dimensão, através de técnicas de compressão de dados [12]. Procura-se assim reduzir o número total de bits a transmitir.

A sequência binária produzida pelo codificador de fonte é colocada na entrada do codificador de canal (*channel encoder*), o qual adiciona redundância à mensagem para fazer face aos erros causados pela passagem no canal de transmissão ruidoso. Os bits redundantes são calculados

[1]Caso o sinal a transmitir já esteja na forma digital, esta acção não é realizada.

em função dos bits de mensagem e desta forma, espera-se conseguir transmissão isenta de erros, sobre o canal ruidoso.

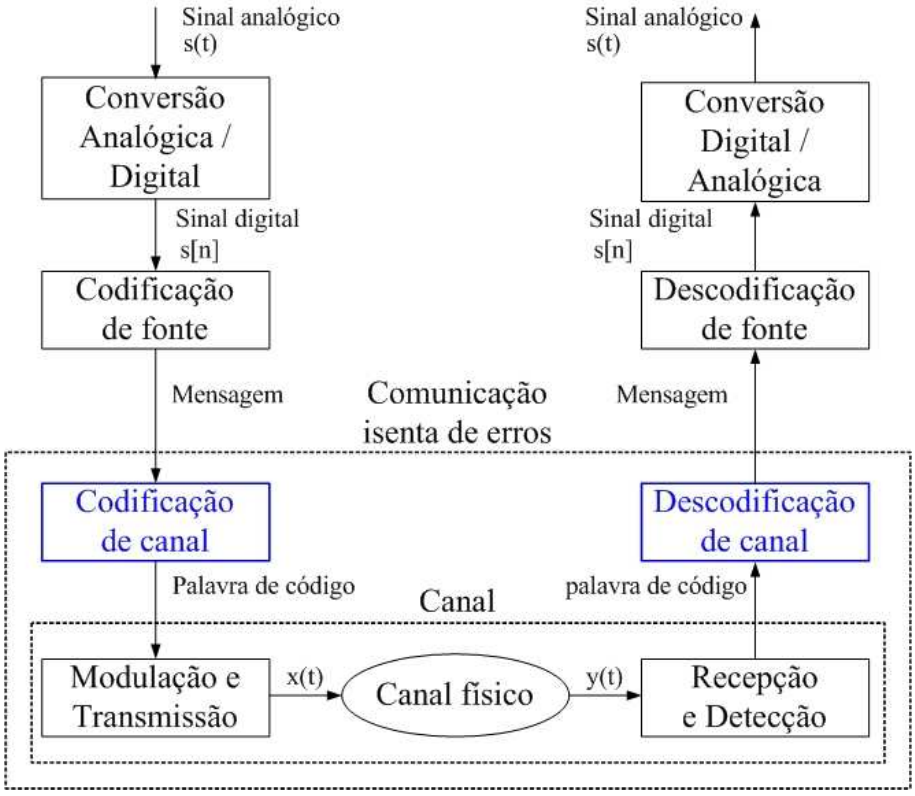


Figura 1: Diagrama de blocos do processo de comunicação.

A figura 2 apresenta o enquadramento da codificação de canal no diagrama geral do processo de comunicação. Nesta figura, mensagem designa a sequência binária produzida pelo codificador de fonte, ou pela saída do ADC, na ausência de codificador de fonte. O codificador de canal recebe a mensagem binária e produz outra sequência binária, constituída por palavras de código. As palavras de código são enviadas para o canal físico de transmissão, após modulação. O descodificador de canal recebe a palavra de código (eventualmente com erros) e procura recuperar a mensagem transmitida.

Neste documento tratam-se códigos de canal, os processos de codificação e de descodificação, sem detalhar aspectos relativos à constituição do canal físico e ao tipo de modulação digital utilizado na transmissão sobre este. O canal físico bem como os processos de modulação e desmodulação são representados através de modelo probabilístico. A proveniência da mensagem a entregar ao codificador de canal é irrelevante, no que respeita ao estudo dos códigos. Assim, neste documento não será considerada a origem das mensagens apresentadas na entrada do codificador de canal.

2.1 Caracterização de canal

Nesta secção, caracterizam-se os canais de comunicação e apresentam-se as principais causas da existência de erros na comunicação. Abordam-se as características dos canais físicos e lógicos.

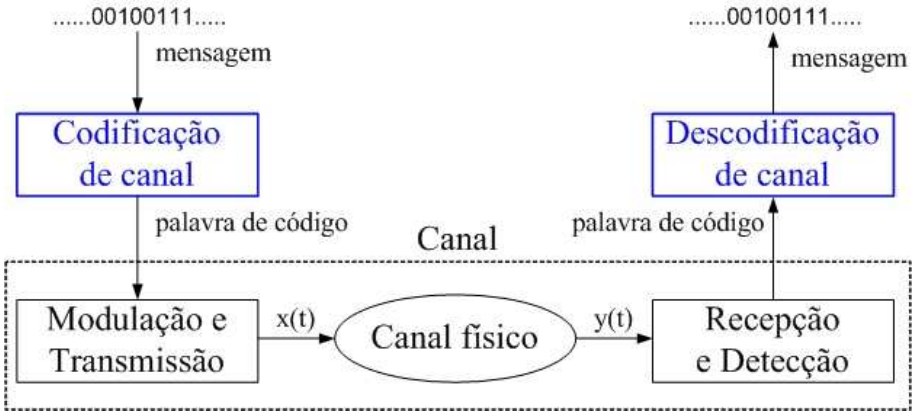


Figura 2: Enquadramento da codificação de canal no processo de comunicação.

2.1.1 Canal físico

A correcção e/ou detecção de erros é necessária devido aos erros introduzidos no canal físico de transmissão. Geralmente, o canal é inacessível e como tal não é possível alterar características deste para evitar ou minimizar a existência de erros. Os erros são causados essencialmente pela conjugação dos seguintes factores: atenuação do sinal transmitido, interferência e ruído. O sinal contínuo $x(t)$ colocado na entrada do canal físico é sujeito a estes factores, de tal forma que no outro extremo do canal se tem um sinal distorcido $y(t)$. A detecção sobre $y(t)$ pode conduzir a bits errados. A figura 3 exemplifica o sinal na entrada $x(t)$ e duas versões possíveis para o sinal de saída, designadas por $y_1(t)$ e $y_2(t)$. Verifica-se que o sinal $y_2(t)$ difere mais do original do que o sinal $y_1(t)$. Desta forma, é mais provável ocorrerem erros na detecção sobre $y_2(t)$ do que sobre $y_1(t)$.

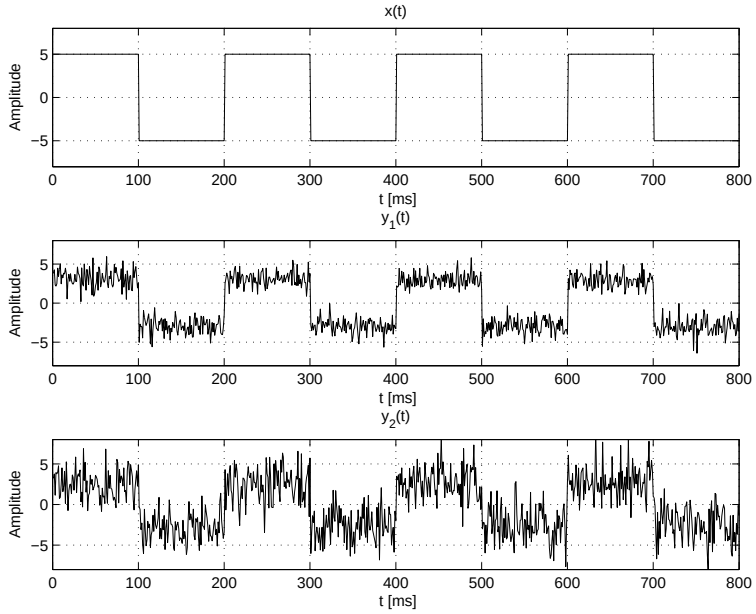


Figura 3: Exemplos de distorções sofridas na passagem do sinal pelo canal. Para o sinal de entrada $x(t)$ (código de linha) são apresentadas duas versões com distorção: $y_1(t)$ e $y_2(t)$.

O modelo AWGN (*Aditive White Gaussian Noise*) [7, 10], apresentado na figura 4, é normalmente utilizado para representar o comportamento de canais físicos. Ao sinal eléctrico (analógico) colocado na entrada do canal, é-lhe adicionado ruído branco, com distribuição de

amplitude gaussiana. O valor β representa a atenuação imposta sobre $x(t)$. O modelo de ruído

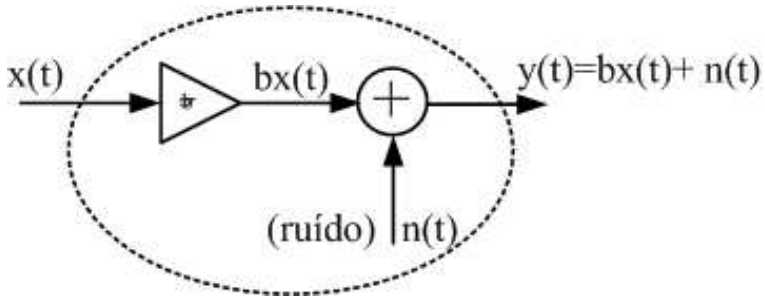


Figura 4: Representação do canal físico: modelo AWGN (*Aditive White Gaussian Noise*).

branco, com densidade espectral plana ($S_n(f) = \eta$), justifica-se pelo facto de que a potência do ruído afecta de igual forma todas as frequências. A distribuição de amplitude deve-se ao teorema do limite central [16].

2.1.2 Modelo de canal

Dado que se realiza comunicação binária, o canal pode ser analisado através de modelo discreto assente em variável aleatórias discretas e binárias. A figura 5 apresenta o modelo genérico de canal discreto binário, no qual a entrada e a saída são representadas por variáveis aleatórias discretas binárias. Neste modelo assume-se que o canal não tem memória, ou seja, a trans-

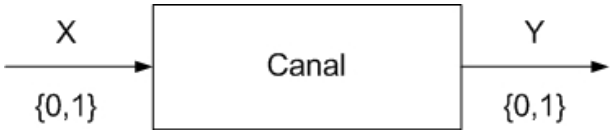


Figura 5: Modelo genérico de canal discreto binário.

missão de um bit é independente das transmissões dos bits anteriores. Tendo em conta que a probabilidade de errar o bit 0 é igual à probabilidade de errar o bit 1, obtém-se o modelo canal binário simétrico (*BSC - Binary Symmetric Channel*). O BSC tem a representação apresentada na figura 6, em que X e Y são variáveis aleatórias binárias que representam a entrada e a saída do canal, respectivamente. A probabilidade de transmitir um bit e detectar o outro designa-se por α. Este modelo probabilístico representa o comportamento conjunto dos blocos

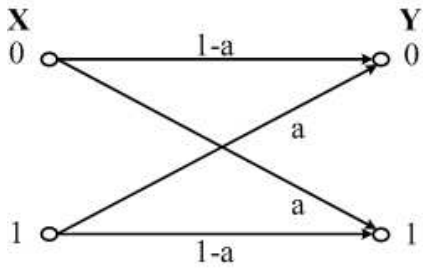


Figura 6: Representação do canal discreto binário: modelo BSC-*Binary Symmetric Channel*.

"modulação e transmissão", "canal físico"e "recepção e detecção", tal como apresentado na figura 2. Assim, as componentes do processo de comunicação apresentadas na figura 2, passam a ser representadas pelo diagrama da figura 7. A matriz estocástica

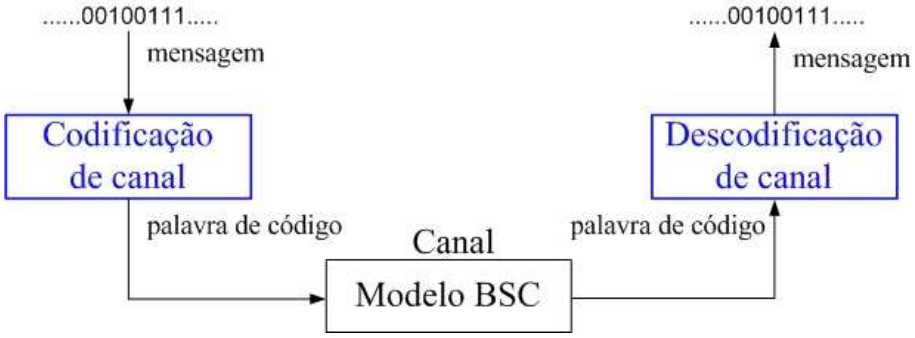


Figura 7: Diagrama do processo de comunicação, apresentado na figura 2, usando modelo probabilístico.

$$\mathbf{P} = \begin{bmatrix} 1-\alpha & \alpha \\ \alpha & 1-\alpha \end{bmatrix}, \tag{1}$$

define as probabilidades de transição $p(y|x)$ associadas ao canal. Na diagonal principal de $\mathbf{P}$ constam as probabilidades de não existir erro na transmissão. As probabilidades de transição $p(y|x)$ são apresentadas na tabela 1.

p(y|x)	Y=0	Y=1
X=0	1-α	α
X=1	α	1-α

Tabela 1: Probabilidades de transição p(y|x), no BSC.

A presença do ruído no canal físico é reflectida no valor atribuído à probabilidade α. Existe erro de transmissão sobre o BSC, quando se transmite determinado bit e se detecta o outro. Desta forma, a probabilidade de erro (P_e) no BSC é dada por

$$\begin{aligned} P_e &= p(Y=0, X=1) + p(Y=1, X=0) \\ &= p(Y=0|X=1)p(X=1) + p(Y=1|X=0)p(X=0) \\ &= \alpha p(X=1) + \alpha p(X=0) \\ &= \alpha, \end{aligned} \tag{2}$$

verificando-se que esta é independente da distribuição de probabilidades de X. A taxa de bits errados, após transmissão pelo canal, depende do valor de P_e. O indicador formal desta taxa de erros designa-se por BER (*Bit Error Rate*). O BER é normalmente utilizado para estabelecer a qualidade de serviço (QOS-*Quality of Service.*) e o tipo de serviço que a utilizar sobre o canal. A transmissão digital de sinais áudio é mais tolerante a erros do que a transmissão de dados. Desta forma, na transmissão de áudio é tolerável a utilização de canais com BER superior aos utilizados para transmissão de dados.

2.2 Teorema da codificação de canal e capacidade de canal

A probabilidade de erro no canal determina a capacidade C de transferência de informação no canal. O segundo teorema de Shannon[2][3, 9, 13], também designado de teorema da codificação

[2]Claude E. Shannon (1916-2001) `http://turnbull.mcs.st-and.ac.uk/~history/Mathematicians/Shannon.html`

de canal, afirma que:

Dada a capacidade C do canal, existe uma técnica de codificação tal que a informação pode ser transmitida no canal a um ritmo R=C, com probabilidade de erro arbitrariamente pequena. Se R>C, não é possível transmitir sem erros.

O teorema da codificação de canal estabelece que a capacidade de canal é o majorante para o ritmo de transferência de informação, com probabilidade de erro arbitrariamente pequena. Estabelece ainda que esta probabilidade de erro é atingível para qualquer ritmo abaixo da capacidade. No entanto, não define como se devem estabelecer os códigos. O trabalho de Hamming[3] consiste em algoritmos para estabelecer esses códigos [6]. Para uma prova do teorema deve consultar [3, 9, 13].

No caso do BSC, a capacidade de transferência de informação, medida em bits/símbolo é dada por

$$C_{\text{BSC}} = 1 + \alpha \log 2(\alpha) + (1-\alpha)\log_2(1-\alpha), \tag{3}$$

em que α é a probabilidade de erro do canal. A figura 8 ilustra esta capacidade de transferência de informação, em função da probabilidade de erro. Verifica-se que o caso mais desfavorável relativo à transferência de informação é atingido com probabilidade de erro $\alpha = 0,5$. Nesta situação, temos que o bit recebido a partir do canal tem tanta probabilidade de estar certo como de estar errado. Assim, neste caso particular, a quantidade de informação transmitida é nula. Note-se que no caso extremo de $\alpha = 1$, temos um canal determinista porque erra sempre o bit transmitido. Assim a quantidade de informação transmitida é máxima e iguala a situação em que $\alpha = 0$.

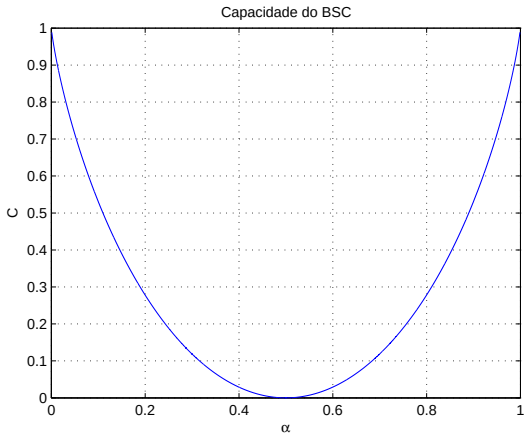


Figura 8: Capacidade de transferência de informação do BSC, em função da probabilidade de erro α.

3 Códigos de codificação de canal

Nesta secção apresentam-se os códigos utilizados na codificação de canal, bem como as suas propriedades e características. Consideram-se os códigos de bloco, nos quais a mensagem e a palavra de código consistem em vectores (blocos) de bits.

[3]Richard W. Hamming (1915-1998) `http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Hamming.html`

3.1 Caracterização do codificador e descodificador

Seja $\mathbf{m} = [m_0\ m_1\ \ldots\ m_{k-1}]$ o vector mensagem com k bits e $\mathbf{c} = [c_0\ c_1\ \ldots\ c_{n-1}]$ a palavra de código com n ($> k$) bits. Existem 2^k mensagens e palavras de código diferentes. A cada mensagem corresponde uma e uma só palavra de código. Essa palavra de código contém os bits de mensagem e os bits de paridade.

3.1.1 Codificador

O codificador de canal consiste numa função que realiza o mapeamento entre a mensagem e a respectiva palavra de código:

$$\text{Cod} : \{0,1\}^k \rightarrow \{0,1\}^n. \tag{4}$$

O codificador calcula

$$q = n - k, \tag{5}$$

bits redundantes em função dos bits da mensagem. Os bits redundantes (também designados por bits de paridade) são concatenados aos de mensagem. Os bits redundantes são calculados em função dos bits de mensagem, de forma a possibilitar ao descodificador a detecção e correcção de erros.

A forma como os bits de paridade são associados aos bits da mensagem classifica o código como sistemático ou não sistemático. Na forma sistemática, os bits redundantes são concatenados no início ou no final dos bits de mensagem. Na forma não sistemática, os bits redundantes são entrelaçados com os bits de mensagem. Designando por $\{b_0, b_1, \ldots, b_{q-1}\}$ os q bits redundantes, apresentam-se alguns exemplos destas formas:

- sistemática
  $\mathbf{c} = [m_0\ m_1\ \ldots\ m_{k-1}\ b_0\ b_1\ \ldots\ b_{q-1}]$,
  $\mathbf{c} = [b_0\ b_1\ \ldots\ b_{q-1}\ m_0\ m_1\ \ldots\ m_{k-1}]$;

- não sistemática
  $\mathbf{c} = [m_0\ b_0\ b_1\ m_1\ \ldots\ m_{k-1}\ \ldots\ b_2\ \ldots\ b_{q-1}]$,
  $\mathbf{c} = [b_0\ b_1\ m_0\ b_2\ m_1 \ldots\ m_{k-1}\ \ldots\ b_3\ \ldots\ b_{q-1}]$.

A introdução de bits redundantes conduz ao afastamento entre as 2^k palavras de código, tal como se ilustra na figura 9, com $k = 2$ e $n = 3$.

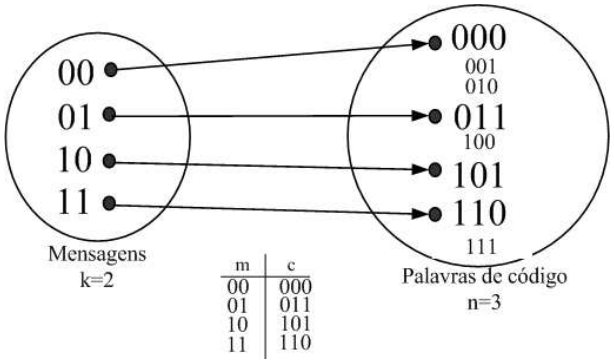


Figura 9: Mapeamento entre mensagens e palavras de código, para o código (3,2).

3.1.2 Descodificador

O descodificador realiza a sequência de acções:

1. recebe a palavra $\mathbf{y}$ (possivelmente com erros);
2. estima a palavra de código $\mathbf{c}$ que lhe deu origem;
3. estima a mensagem $\mathbf{m}$ correspondente.

A figura 10 ilustra as acções do codificador, descodificador e do canal. A mensagem $\mathbf{m}$ é transformada na palavra de código $\mathbf{c}$. Esta é enviada através do canal, obtendo-se

$$\mathbf{y} = \mathbf{c} \oplus \mathbf{e} \tag{6}$$

na saída deste, em que $\oplus$ representa a soma binária, sem arrasto (operação XOR) e $\mathbf{e}$ representa o padrão de erro adicionado à palavra de código. O padrão de erro consiste num vector binário que tem o valor 1 nas posições onde se introduz erro na palavra. Por exemplo, considerando que sobre a palavra $\mathbf{c} = [011011]$ se erra o primeiro e último bit, temos que o padrão de erro é $\mathbf{e} = [100001]$. Nesta situação a palavra recebida seria $\mathbf{y} = [111010]$. Caso $\mathbf{e}$ seja diferente do vector nulo, a palavra recebida tem erros. O descodificador procura estimar a palavra de código transmitida e a respectiva mensagem $\hat{\mathbf{m}}$.

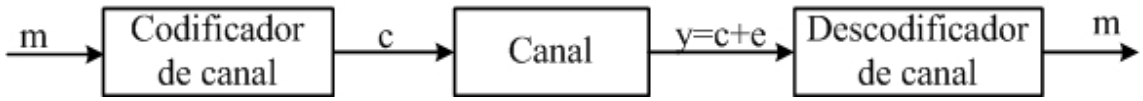


Figura 10: Sequência de acções realizada pelo codificador e pelo descodificador de canal.

Caso a palavra recebida $\mathbf{y}$ não pertença ao código, então ocorreu um ou mais erros na transmissão. Para lidar com a existência de erros, o descodificador funciona num dos modos:

- detecção;
- correcção;
- detecção e correcção.

No modo de detecção, quando o descodificador detecta que existe erro na palavra recebida, reporta esse erro e envia um pedido de retransmissão. No modo de correcção, sempre que detectada a presença de um (ou mais) erro(s) procura-se corrigir esse(s) erro(s). O primeiro modo designa-se por ARQ (*Automatic Repeat ReQuest*), enquanto que o segundo toma o nome de FEC (*Forward Error Correction*). É ainda possível funcionar em modo misto (detecção e correcção) procurando corrigir os erros para os quais existe redundância suficiente para o fazer, e pedir a retransmissão da palavra de código sempre que forem detectados erros para os quais não há capacidade de correcção.

3.2 Códigos de bloco

As características dos códigos, nomeadamente as capacidades de detecção e correcção de erros determinam o modo de funcionamento do descodificador. Nesta secção analisam-se as características dos códigos de bloco, considerando que existem k bits de mensagem e n bits na palavra de código: estes códigos são referidos como (n,k). O codificador mapeia esse bloco noutro de dimensão n, designado de palavra de código.

3.2.1 Características

De forma a analisar as características dos códigos é necessário ter em conta a definição de distância de Hamming entre duas palavras. A distância de Hamming (dH) entre duas palavras de código c_i e c_j define-se como o número de bits que varia entre essas duas palavras. Por exemplo, para $\mathbf{c_1}$ = [0 1 1 0 1 1] e $\mathbf{c_2}$ = [1 1 0 0 1 0], temos dH(c_1,c_2)=3. A distância de Hamming é sempre não negativa, sendo nula apenas no caso em que é avaliada entre uma palavra e ela própria: dH(c_i,c_i)=0. Relativamente aos códigos, as características a considerar são as seguintes [10, 18]:

- **Code rate (ritmo)** $R = \frac{k}{n} = \frac{k}{k+q}$; é a medida de eficiência do código, porque representa o quociente do número de bits de informação sobre o número total de bits transmitidos.
- **Distância mínima** ($\mathrm{d_{min}}$) é a menor distância de Hamming entre duas quaisquer palavras do código; depende do número de bits redundantes $q = n - k$ [18], tal que $\mathrm{d_{min}} \leq q + 1$.
- **Capacidade de detecção** - detecta todos os padrões até l erros, com $l \leq \mathrm{d_{min}} - 1$.
- **Capacidade de correcção** - corrige todos os padrões até t erros, com $t \leq \lfloor \frac{d_{min}-1}{2} \rfloor$, em que $\lfloor x \rfloor$ representa o maior número inteiro, não superior a x.
- **Capacidade de detecção e correcção** - detecta até l erros e corrige até t erros com $\mathrm{d_{min}} \geq l + t + 1$ e $l > t$.

A distância de Hamming é uma medida aplicável a duas quaisquer palavras do código e traduz o número de bits em que estas diferem. Considerando duas palavras de código c_1 e c_2, a distância de Hamming entre as mesmas é dada pelo número de bits a 1, presentes na palavra binária $c = c_1 \oplus c_2$. A distância mínima sendo a menor das distâncias de Hamming entre todas as palavras de código, é uma característica do código e determina as suas capacidades de detecção e correcção de erros.

3.2.2 Capacidades de detecção e correcção de erros

A capacidade de detecção de erros define-se como o número máximo de bits em erro que é possível detectar numa palavra de código. Esta capacidade é representada pela quantidade $l(< n)$, sendo n a dimensão da palavra de código. A capacidade de correcção de erros corresponde ao máximo número de bits que é possível corrigir, numa palavra de código. Esta capacidade é representada pela quantidade $t(< l)$. Para realizar correcção de erros, é necessário realizar previamente a detecção e localização dos mesmos. Assim, o número de bits que se consegue corrigir é sempre inferior ao número de bits em erro que se consegue detectar.

A capacidade de detecção de erros em função da distância mínima é ilustrada na figura 11. As palavras c_1 e c_2 diferem entre si em 3 bits. Se considerarmos que, de entre todas as palavras de código, estas são as mais semelhantes entre si temos que o código tem $d_{min} = 3$. Caso seja transmitida a palavra c_1 ou a palavra c_2 e seja recebida outra configuração o descodificador detecta a presença de erros. Desta forma, verifica-se que com $d_{min} = 3$, o código tem capacidade de detecção de 1 e 2 bits em erro, detectando até 2 bits em erro na palavra. A figura 11 também serve para ilustrar a capacidade de correcção. Caso se introduza 1 erro sobre a palavra c_1 obtemos a configuração `0010` que não é uma palavra de código. Caso esta palavra seja recebida, o descodificador vai escolher a palavra mais próxima (c_1), corrigindo o erro. Mas, considerando que se envia a mesma palavra c_1 e que esta sofre 2 erros, temos que o descodificador recebe `0000`. Nesta situação, a palavra de código mais semelhante é c_2. Assim, existe erro na estimação

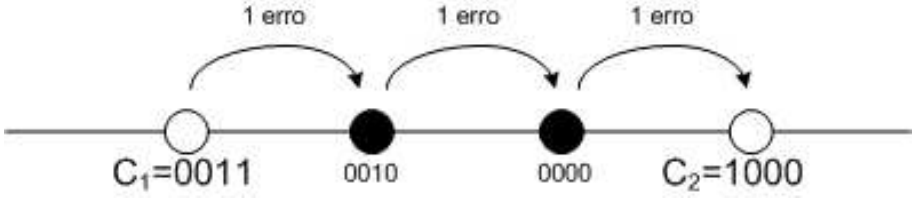


Figura 11: Ilustração das capacidades de detecção e correcção de erros, com $d_{min} = 3$.

da palavra de código e verifica-se que o código não tem capacidade de correcção de 2 bits em erro, tendo apenas a possibilidade de corrigir 1 bit em erro.

Considere-se agora a situação em que as palavras de código c_1 e c_2 diferem entre si 4 bits, tal como apresentado na figura 12. Neste caso é possível detectar até 3 bits em erro, sendo possível corrigir apenas 1 bit em erro. Note-se que se a palavra c_1 for transmitida, sendo-lhe adicionados dois erros, obtém-se a configuração `00000`, a qual está à mesma distência de c_1 e c_2. Assim, é impossível realizar a correcção com certeza absoluta de acerto. Conclui-se, desta forma, que um código com $d_{min} = 4$ não tem capacidade de corrigir 2 bits em erro, tendo apenas capacidade de correcção até $t = 1$ bit em erro por palavra de código.

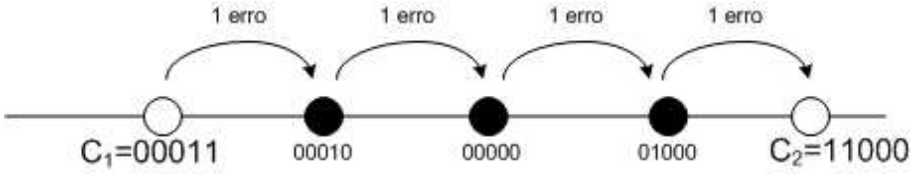


Figura 12: Ilustração das capacidades de detecção e correcção de erros, com $d_{min} = 4$.

A tabela 2 apresenta as capacidades de detecção, correcção e detecção e correcção simultâneas, em função da distância mínima, para uma gama de valores. Por análise da tabela

d_{min}	**Detecção (l)**	**Correcção (t)**	**Det. e Corr. simultâneas (l,t)**
1	0	0	Não tem
2	1	0	Não tem
3	2	1	Não tem
4	3	1	(2,1)
5	4	2	(3,1)

Tabela 2: Capacidades de detecção e correcção de erros, em função da distância mínima.

verifica-se que:

- apenas para $d_{min} \geq 4$ é que se torna possível a detecção e a correcção em simultâneo;
- a capacidade de detecção é sempre superior à capacidade de correcção;
- um código com $d_{min} = 1$ não tem capacidade para detectar erros; por exemplo, considerando as 8 palavras do código binário natural a 3 bit, verifica-se que qualquer alteração de um bit numa palavra vai produzir outra palavra que pertence ao código; este erro é indetectável (as palavras de código são excessivamente semelhantes entre si).

As capacidades de detecção e correcção são obtidas à custa da introdução de redundância e dependem da distância mínima do código. Aumentar a distância mínima melhora as capacidades de detecção e correcção, mas em contrapartida diminui a eficiência do código. Os critérios de desenho dos códigos de codificação de canal são:

- dado o R maximizar d_{min};
- dada a d_{min} minimizar R.

O desenho de códigos eficientes constitui um problema complexo. Em seguida, são analisados os códigos de repetição e de bit de paridade par, para os quais a abordagem de desenho é relativamente simples. Serão ainda abordados os códigos lineares enquanto sub-classe de códigos, caracterizados por possuirem construção fácil, baixa complexidade de codificação e de descodificação e capacidades razoáveis de detecção e correcção.

Os códigos de bloco (n,k) também podem ser representados na forma (n,M,d), sendo n o comprimento das palavras, $M = 2^k$ o número de palavras de código e d a distância mínima do código [18].

3.2.3 Código de repetição (3,1)

A repetição é uma forma simples de introduzir redundância na mensagem. Considerando mensagens com k=1 bit e introduzindo dois bits de redundantes iguais que constituem repetição da mensagem, tem-se o código (3,1) apresentado na tabela 3.

Mensagem	**Palavra de Código**
0	000
1	111

Tabela 3: Mensagens e palavras de código para o código de repetição (3,1).

A figura 13 ilustra o mapeamento entre o espaço das mensagens e das palavras de código. Note-se o afastamento entre as palavras do código.

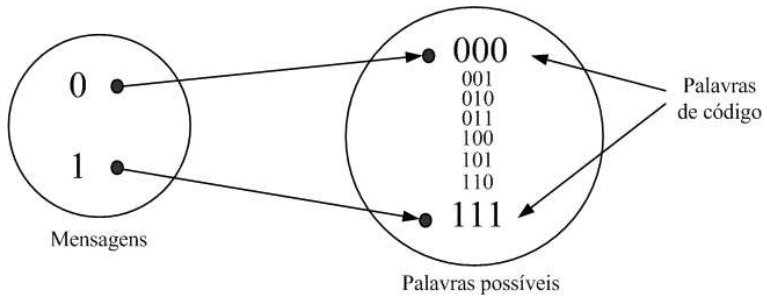


Figura 13: Código de repetição (3,1): mapeamento entre mensagens e palavras de código.

A descodificação deste código é realizada por maioria, ou seja, se a maioria dos bits da palavra de código recebida valer 1, então descodifica-se a mensagem 1, caso contrário, descodifica-se a mensagem 0. Este código tem d_{min}=3 e como tal:

- detecta todos os erros de 1 e 2 bit;
- corrige todos os erros de 1 bit.

A figura 14 ilustra estas capacidades de detecção e correcção de erros, evidenciando o afastamento entre as palavras do código. Este código pode funcionar nos modos de FEC e ARQ.

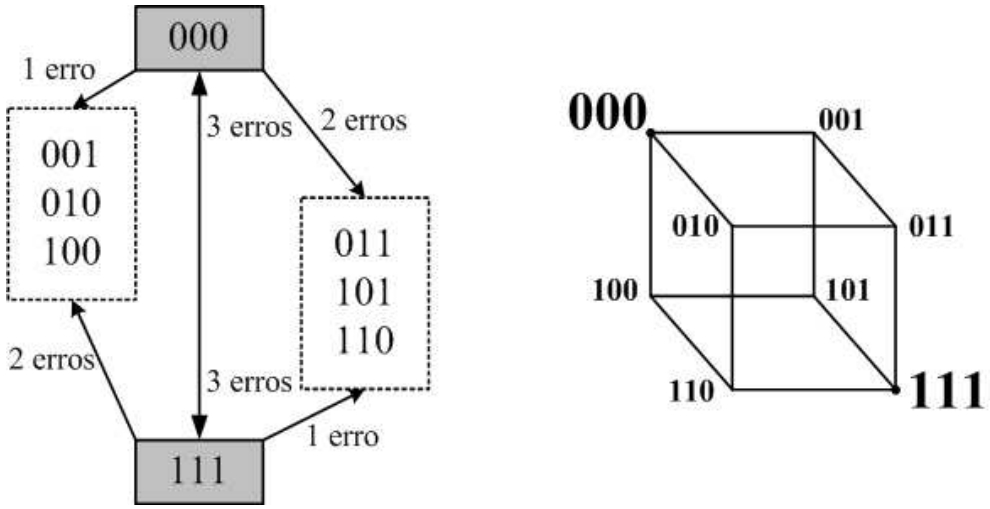


Figura 14: Ilustração das capacidades de detecção e correcção de erros do código de repetição (3,1).

Considerando que se utiliza este código sobre um BSC com $P_e = \alpha = 10^{-5}$ tem-se que a probabilidade de errar 1 bit, sobre uma palavra de 3 bits, é dada por

$$P(1,3) = C_1^3\ \alpha^1(1-\alpha)^2 = \frac{3!}{2!1!}\ \alpha(1-\alpha)^2 = 3\alpha - 6\alpha^2 + 3\alpha^3 \approx 3\times 10^{-5}, \tag{7}$$

em que C_1^3 representa combinações de três um a um. A probabilidade de errar 2 bits é

$$P(2,3) = C_2^3\ \alpha^2(1-\alpha)^1 = \frac{3!}{1!2!}\ \alpha^2(1-\alpha) = 3\alpha^2 - 3\alpha^3 \approx 3\times 10^{-10}, \tag{8}$$

Finalmente, a situação extrema de errar os 3 bits da palavra ocorre com probabilidade

$$P(3,3) = C_3^3\ \alpha^3(1-\alpha)^0 = \frac{3!}{0!3!}\ \alpha^3 = \alpha^3 = 10^{-15}, \tag{9}$$

concluindo-se que $P(3,3) \ll P(2,3) \ll P(1,3)$. Verifica-se que a capacidade de correcção até 1 bit errado é adequada nesta situação.

3.2.4 Código bit de paridade par (3,2)

O código bit de paridade par (3,2), consiste em adicionar um bit de paridade no final da mensagem. Este bit é a soma módulo 2 dos bits da mensagem obtendo-se assim a palavra de código $\mathbf{c} = [m_0\ m_1\ \ m_0 \oplus m_1]$. A tabela 4 apresenta as palavras de código. Este código tem d_{min}=2 e detecta a presença de 1 e 3 bits errados[4], não tem capacidade de correcção e como tal não pode ser utilizado no modo FEC. A descodificação é realizada recalculando a paridade da mensagem recebida, comparando-a com a paridade transmitida; se forem iguais não se detectam erros, caso contrário são detectados 1 ou 3 erros, na palavra recebida. A figura 15 ilustra a disposição relativa das palavras de código, evidenciando que $d_{min} = 2$, uma vez que entre quaisquer duas palavras do código, é sempre necessário percorrer duas arestas do cubo. Por outro lado, verifica-se que a existência de 3 erros sobre qualquer palavra de código é sempre detectável, uma vez que resulta numa palavra que não pertence ao código.

[4]O valor d_{min}=2 garante a detecção de todos os erros de 1 bit, para qualquer código. Neste código, é ainda possível detectar 3 bits errados.

Mensagem	Palavra de Código
00	000
01	011
10	101
11	110

Tabela 4: Mensagens e palavras de código para o código bit de paridade par (3,2).

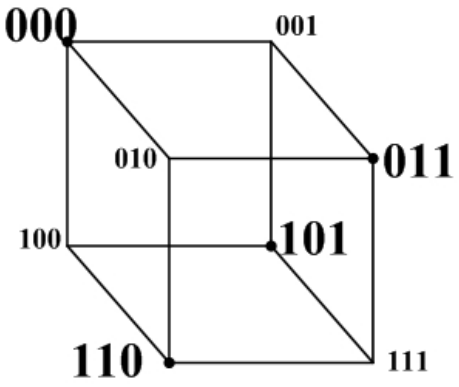


Figura 15: Palavras do código de bit de paridade par (3,2).

4 Códigos lineares de bloco

Tal como constatado anteriormente, o desenho de códigos eficientes é um problema complexo. Procura-se maximizar a distância mínima do código (d_{min}), tendo como constrição o *code rate* R. Em alternativa, podemos maximizar R com constrição d_{min}. São também problemas adicionais a memória ocupada e complexidade do codificador e do descodificador.

Recorrendo aos conceitos de estrutura algébrica e espaço vectorial [4, 8, 15, 18] definem-se os códigos lineares. As palavras de código são elementos de determinado sub-espaço vectorial. Obtêm-se assim os **códigos lineares de bloco** [2, 7, 17, 18]. Designam-se de **bloco** porque todas as palavras têm a mesma dimensão e **lineares** porque:

- o vector nulo é palavra do código $\mathbf{c} = [0\ 0\ 0 \ldots 0]$;
- a soma modular de duas palavras do código $c_k = c_i \oplus c_j$ é ainda uma palavra do código.

Os códigos lineares são um sub-conjunto de todos os códigos, apresentando as vantagens de requererem pouca memória e os codificadores e descodificadores são constituídos por operações simples.

4.1 Características

O peso de Hamming ($\mathbf{w}$) de uma palavra define-se como o número de bits não nulos nessa palavra. Sejam c_i e c_j duas palavras distintas de um código linear de bloco. A d_{min} do código é dada por

$$d_{min} = \min_{i \neq j} dH(c_i, c_j). \tag{10}$$

Dado que o código é linear, tem-se que a soma modular de duas palavras $c_k = c_i \oplus c_j$ é ainda outra palavra do código, diferente do vector nulo. Desta forma, a d_{min} do código é dada por

$$d_{min} = \min\ \mathbf{w}(c_k), \tag{11}$$

sendo c_k qualquer palavra de código diferente do vector nulo. O peso de Hamming desta palavra corresponde ao número de bits em que c_i e c_j diferem.

Os códigos de repetição e de bit de paridade par também são lineares. As tabelas 5 e 6 apresentam as palavras destes códigos e os respectivos pesos de Hamming, através dos quais se confirma que:

- o código de repetição (3,1) tem $\mathrm{d_{min}} = 3$, $l = 2$ e $t = 1$.
- o código de bit de paridade par (3,2) tem $\mathrm{d_{min}} = 2$, $l = 1$ e $t = 0$.

Mensagem	**Palavra de Código**	**Peso de Hamming**
0	000	0
1	111	3

Tabela 5: Mensagens, palavras de código e respectivo peso de Hamming para o código de repetição (3,1).

Mensagem	**Palavra de Código**	**Peso de Hamming**
00	000	0
01	011	2
10	101	2
11	110	2

Tabela 6: Mensagens, palavras de código e respectivo peso de Hamming para o código bit de paridade par (3,2).

4.1.1 Códigos de Hamming

Os códigos de Hamming[5][2, 3, 6, 7] constituem uma família de códigos lineares de bloco, desenhada com o critério de possuírem sempre $\mathrm{d_{min}} = 3$, corrigindo todos os erros de 1 bit. A motivação deste critério de desenho está no facto de que sobre um BSC, a probabilidade de errar 2 bits na mesma palavra de código é muito menor do que a probabilidade de errar apenas 1 bit: $P(2, n) \ll P(1, n)$. Os códigos de Hamming são definidos por um parâmetro inteiro r (≥ 2) tal que: $(n, k) = (2^r - 1, 2^r - 1 - r)$. A tabela 7 apresenta as dimensões das palavras de código em função do valor do parâmetro r. O código (3,1), obtido com $r = 2$ é o código de repetição mencionado anteriormente. Por exemplo com $r = 3$ tem-se $(n, k) = (7, 4)$ e estabelecendo as equações de paridade

$$\begin{aligned} b_0 &= m_1 \oplus m_2 \oplus m_3, \\ b_1 &= m_0 \oplus m_1 \oplus m_3, \\ b_2 &= m_0 \oplus m_2 \oplus m_3, \end{aligned} \tag{12}$$

obtêm-se as palavras de código na forma sistemática

$$\mathbf{c} = [m_0\ m_1\ m_2\ m_3\ b_0\ b_1\ b_2]. \tag{13}$$

r	(n,k)
2	(3,1)
3	(7,4)
4	(15,11)
5	(31,26)
6	(63,57)
...	...

Tabela 7: Dimensões do código de Hamming em função do parâmetro de desenho r.

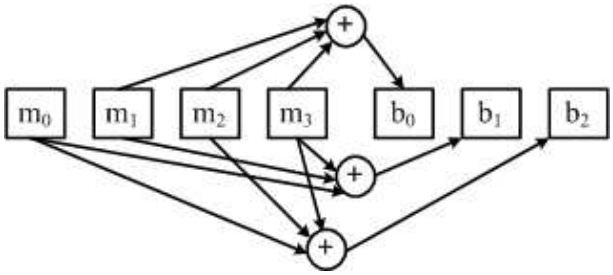


Figura 16: Bits de paridade no código Hamming (7,4).

A figura 16 ilustra os bits de paridade utilizados no código Hamming (7,4). Verifica-se que cada palavra de código pode então ser escrita na forma matricial

$$\mathbf{c} = m\mathbf{G} = m\left[I_4 \mid P\right] = \left[\begin{array}{cccc} m_0 & m_1 & m_2 & m_3 \end{array}\right] \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]. \tag{14}$$

A matriz $\mathbf{P}$ de dimensões $k \times q$ designa-se por sub-matriz geradora de paridade; cada coluna de $\mathbf{P}$ estabelece uma equação de paridade, ou seja, cada um dos bits $\{b_0, b_1, b_2\}$ de (12). A matriz $\mathbf{G}$ de dimensões $k \times n$ designa-se de matriz geradora do código, uma vez que estabelece todas as palavras do código. Cada linha de $\mathbf{G}$ é uma palavra do código. Todas as palavras do código são obtidas por combinação linear das linhas de $\mathbf{G}$, os coeficientes da combinação linear são os bits da mensagem. A tabela 8 apresenta as $2^4 = 16$ mensagens, as palavras do código e o respectivo peso de Hamming. Verifica-se que, à excepção do vector nulo, o menor peso de Hamming de todas as palavras vale 3, logo o código tem $\mathrm{d_{min}} = 3$. Note-se que este é um código de Hamming. Estabelecendo outra sub-matriz geradora de paridade $\mathbf{P}$, com pelo menos dois bits a 1 por cada linha, obtém-se outro código de Hamming. As matrizes geradoras

$$\mathbf{G_1} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right] \quad \text{e} \quad \mathbf{G_2} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right] \tag{15}$$

também estabelecem códigos de Hamming (7,4).

[5]Richard W. Hamming (1915-1998) `http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Hamming.html`

Mensagem	Palavra de Código	Peso de Hamming
0000	0000000	0
0001	0001111	4
0010	0010101	3
0011	0011010	3
0100	0100110	3
0101	0101001	3
0110	0110011	4
0111	0111100	4
1000	1000011	3
1001	1001100	3
1010	1010110	4
1011	1011001	4
1100	1100101	4
1101	1101010	4
1110	1110000	3
1111	1111111	7

Tabela 8: Mensagens, palavras de código e peso de Hamming para o código Hamming (7,4), definido em (14).

4.2 Tratamento matricial dos códigos

Os códigos de bloco linear caracterizam-se por ter codificação e descodificação com baixa complexidade, ocupando pouca memória. Dado que as palavras de código são elementos de um sub-espaço vectorial caracterizado pela respectiva matriz geradora, torna-se assim possível gerar todas as palavras a partir de um sub-conjunto de palavras.

4.2.1 Codificação

A codificação consiste em multiplicar o vector mensagem (de dimensões $1 \times k$) pela matriz geradora (de dimensões $k \times n$) do código obtendo a palavra de código

$$\mathbf{c} = \mathbf{mG}, \tag{16}$$

de dimensões $1\times$ n. Na perspectiva vectorial, tem-se que $\mathbf{G}$ é um conjunto de vectores linearmente independentes, ou seja, nenhuma linha de $\mathbf{G}$ pode ser obtida por combinação linear das outras linhas. A matriz $\mathbf{G}$ gera $2^4 = 16$ vectores de um total possível de $2^7 = 128$ e como tal é a base de sub-espaço vectorial. As palavras do código são elementos desse sub-espaço vectorial. Note-se a economia de memória na codificação: para obter as 16 palavras de código, basta armazenar 4 palavras. Considerando o código de Hamming (15,11), com $2^{11} = 2048$ palavras, verifica-se que a matriz geradora é constituída apenas por k=11 palavras. Em termos genéricos, geram-se 2^k palavras de dimensão n, através da combinação linear de k palavras. A matriz geradora $\mathbf{G}$, de dimensões $k \times n$ é constituída por palavras do código de tal forma que cumpra as seguintes condições:

- não possui o vector nulo;
- não existem duas linhas iguais;

- nenhuma linha da matriz pode ser obtida por combinação linear de duas ou mais linhas da matriz.

As matrizes geradores para os códigos de repetição (3,1) e de bit de paridade par (3,2), são

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \qquad \text{e} \qquad \mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \tag{17}$$

respectivamente. A partir destas matrizes, geram-se todas as palavras de código apresentadas nas tabelas 5 e 6. Para o código de repetição (3,1) temos que as palavras de código estão organizadas na forma $c = [m_0\ m_0\ m_0]$ enquanto que para o código de bit de paridade par (3,2) temos $c = [m_0\ m_1\ m_0 \oplus m_1]$.

4.2.2 Descodificação

A descodificação assume uma forma semelhante à codificação, na perspectiva matricial. A palavra de código recebida é também multiplicada por uma matriz, de dimensões $n \times q$, que designaremos por $\mathbf{H^T}$, matriz de teste de paridade. Desta multiplicação obtém-se o vector

$$\mathbf{s} = \mathbf{cH^T}, \tag{18}$$

de dimensões $1 \times q$, designado por <u>síndroma</u>. A matriz $\mathbf{H^T}$ possui a forma $\mathbf{H^T} = \begin{bmatrix} P \\ I_q \end{bmatrix}$ e é ortogonal à matriz geradora

$$\mathbf{GH^T} = [\mathbf{I_k} \quad \mathbf{P}] \begin{bmatrix} P \\ I_q \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \dots & 0 \\ \vdots & \dots & \vdots \\ 0 & \dots & 0 \end{bmatrix}}_{\text{Matriz nula com k×q bits}}. \tag{19}$$

Dado que a palavra de código é escrita na forma $\mathbf{c} = \mathbf{mG}$, tem-se que

$$\mathbf{s} = \mathbf{cH^T} = \mathbf{mGH^T} = \underbrace{[0\ 0\ \dots\ 0]}_{\text{Vector nulo com q bits}} \tag{20}$$

para qualquer palavra do código $\mathbf{c}$. Caso o descodificador receba a palavra $\mathbf{y} = \mathbf{c} + \mathbf{e}$, a qual não pertence ao código, dado que $\mathbf{e}$ é o padrão de erro adicionado à palavra de código $\mathbf{c}$, o valor de $\mathbf{s}$ é não nulo. Neste caso tem-se

$$\mathbf{s} = \mathbf{yH^T} = (\mathbf{c} + \mathbf{e})\mathbf{H^T} = (\mathbf{mG} + \mathbf{e})\mathbf{H^T} = \underbrace{\mathbf{mGH^T}}_{[0\ 0\ \dots\ 0]} + \mathbf{eH^T} = \mathbf{eH^T}, \tag{21}$$

verificando-se que o síndroma apenas depende do padrão de erro $\mathbf{e}$. A matriz $\mathbf{H^T}$ é construída de acordo com a forma da matriz geradora $\mathbf{G}$. A funcionalidade de $\mathbf{H^T}$ consiste em recalcular os bits de paridade sobre os bits de mensagem e comparar esses bits de paridade com aqueles transmitidos na palavra. Quando a matriz geradora $\mathbf{G}$ assume a forma $\mathbf{G} = [\mathbf{I_k} \quad \mathbf{P}]$, tem-se que $\mathbf{H^T} = \begin{bmatrix} P \\ I_q \end{bmatrix}$. Para o código de Hamming (7,4) apresentado em (12), tem-se

$$\mathbf{H^T} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{22}$$

No caso dos códigos de repetição (3,1) e bit de paridade par (3,2), estas matrizes são

$$\mathbf{H^T} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{e} \qquad \mathbf{H^T} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{23}$$

respectivamente. Verifica-se que o produto das respectivas matrizes geradoras por estas matrizes de teste de paridade, produzem matrizes nulas, de acordo com (19). Cada bit do síndroma representa a verificação realizada sobre os bits de paridade, através da comparação entre o bit de paridade transmitido e o recalculado na recepção. Esta verificação é conseguida pela multiplicação matricial. Sempre que um bit do síndroma seja não nulo tal significa que foi detectado um erro através desse bit de paridade. Assim, o síndroma nulo indica que não se detectam erros. Por exemplo, sempre que o primeiro bit do síndroma for 1, tal indica que foi detectado um erro, através do primeiro bit de paridade.

4.2.3 Tabela de síndromas

Cada linha de $\mathbf{H^T}$ representa um síndroma do código. Associado a cada síndroma, está um padrão de 1 bit em erro. Sempre que esse padrão de erro se verificar sobre a palavra, obtém-se o respectivo síndroma, tal como se constata através de (21). Sabendo que

$$\mathbf{eH^T}, \tag{24}$$

e considerando palavras de código com $n = 7$ bit e tendo em conta o padrão de 1 bit em erro, $e = [1\ 0\ 0\ 0\ 0\ 0\ 0]$ temos

$$[1\ 0\ 0\ 0\ 0\ 0\ 0]\mathbf{H}^T = 1^{\text{a}} \text{ linha de } \mathbf{H}^T.$$

Caso o bit errado seja o segundo, temos

$$[0\ 1\ 0\ 0\ 0\ 0\ 0]\mathbf{H}^T = 2^{\text{a}} \text{ linha de } \mathbf{H}^T$$

e assim sucessivamente para os restantes padrões de 1 bit em erro.

A tabela 9 mostra os síndromas e os respectivos padrões de erro, para o código Hamming (7,4), definido em (14). Ao síndroma nulo, corresponde a ausência de erros detectados. O número de síndromas é dado por $2^q - 1 = 2^3 - 1 = 7$, tantos quantos os padrões de erro de 1 bit. Devido a esta correspondência, os códigos de Hamming são designados de códigos perfeitos [18].

A figura 17 ilustra o processo de descodificação e consulta à tabela de síndromas para correcção. Em função do síndroma obtido, extrai-se o respectivo padrão de erro e soma-se este à palavra de código recebida e a partir desta estima-se a mensagem enviada. Para funcionamento

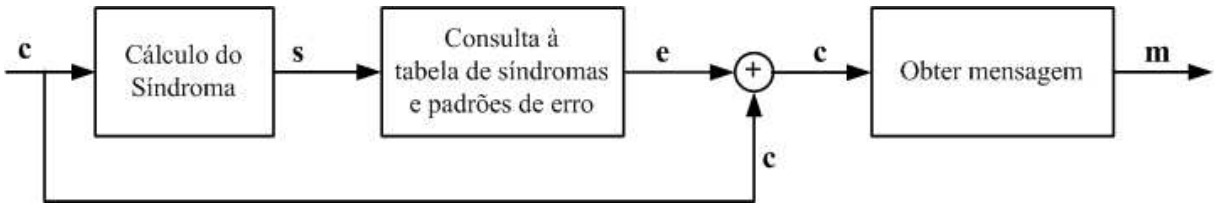


Figura 17: Descodificação baseada em síndroma.

no modo de detecção, basta verificar se o síndroma é ou não nulo. Caso o síndroma seja nulo,

Síndroma (s)	Padrão de erro (e)	Observações
000	0000000	Ausência de erros
011	1000000	1.º bit em erro
110	0100000	2.º bit em erro
101	0010000	3.º bit em erro
111	0001000	4.º bit em erro
100	0000100	5.º bit em erro
010	0000010	6.º bit em erro
001	0000001	7.º bit em erro

Tabela 9: Tabela de síndromas e respectivos padrões de erro, para o código Hamming (7,4).

Figura 18: *Flow-chart* da descodificação baseada em síndroma - detecção de erros.

não são detectados erros. Caso seja não nulo, detecta-se a presença de erros. A figura 18 apresenta o *flow-chart* do algoritmo de detecção de erros, baseado em síndroma. O *flow-chart* correspondente para a correcção de erros é apresentado na figura 19.

Note-se que, à semelhança do codificador, também o descodificador necessita de pouca memória: basta possuir a matriz $\mathbf{H^T}$ e os padrões de erro (ou, em alternativa, a informação de qual o bit errado para cada síndroma).

4.2.4 Exemplos de descodificação

Exemplifica-se a descodificação e a correcção de erros, utilizando as palavras $\mathbf{c} = [0\ 0\ 0\ 0\ 0\ 0\ 0]$ e $\mathbf{c} = [0\ 0\ 0\ 1\ 1\ 1\ 1]$ do código de Hamming (7,4), com matriz $\mathbf{H^T}$ definida por (22). A tabela 10 apresenta exemplos de descodificação, em modo correcção, sobre estas palavras, na presença de 0, 1, 2 e 3 bits errados. A tabela também pode ser utilizada para análise em modo de detecção de erros, considerando a análise da esquerda para a direita até chegar à coluna do cálculo do síndroma. Verifica-se que para as situações em que existe 1 bit errado, a correcção é realizada. Nas situações em que existem 2 bits errados, acima da capacidade de correcção do código, o mecanismo de correcção baseado em síndroma introduz mais um erro e a mensagem estimada

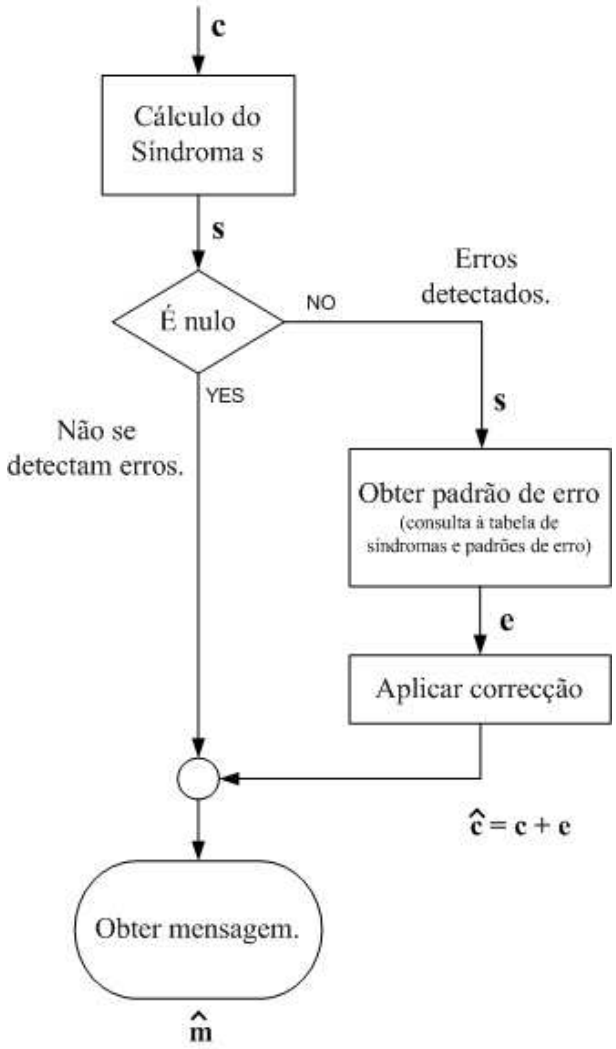


Figura 19: *Flow-chart* da descodificação baseada em síndroma - correcção de erros.

Situação	**m**	**c**	**e**	**y=c+e**	**s**	**ĉ**	**m̂**	**Observação**
0 erros	0001	0001111	0000000	0001111	000	0001111	0001	Ok
0 erros	0000	0000000	0000000	0000000	000	0000000	0000	Ok
1 erro	0001	0001111	1000000	1001111	011	0001111	0001	Ok
1 erro	0000	0000000	0000001	0000001	001	0000000	0000	Ok
2 erros	0001	0001111	0001010	0000101	101	0010101	**0010**	Erro
2 erros	0000	0000000	1100000	1100000	101	1110000	**1110**	Erro
3 erros	0001	0001111	1010100	1011011	010	1011001	**1011**	Erro
3 erros	0000	0000000	1110000	1110000	000	1110000	**1110**	Erro

Tabela 10: Exemplos de descodificação em modo correcção, na presença de erros, para o código Hamming (7,4).

está errada. Caso existam 3 erros, também não é possível descodificar a mensagem correcta. Na última linha da tabela verifica-se que após o erro, obteve-se outra palavra do código. Nesta situação o síndroma calculado é nulo. Note-se que se o descodificador funcionar em modo de detecção de erros, são detectados todos os erros de 1 bit e 2 bit por palavra de código. É ainda possível detectar alguns erros de 3 bit, sempre que o síndroma obtido é não nulo, tal como

acontece no teste realizado na penúltima linha da tabela.

4.2.5 Cálculo da distância mínima

No processo de descodificação as palavras recebidas são multiplicadas por $\mathbf{H^T}$. As palavras de código são ortogonais a $\mathbf{H^T}$ e a multiplicação das palavras de código por $\mathbf{H^T}$ consiste na combinação linear das linhas desta matriz; estas linhas são as que correspondem aos bits com o valor 1 na palavra de código. Conclui-se que a d_{min} do código pode ser obtida a partir das linhas de $\mathbf{H^T}$, como o número mínimo de linhas que é necessário somar para obter o vector nulo [7]. De seguida, apresenta-se um exemplo desta forma de cálculo da distância mínima. Considere-se a matriz de teste de paridade

$$\mathbf{H^T} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{25}$$

Verifica-se que para obter o vector nulo é necessário somar pelo menos três linhas desta matriz[6]. Por exemplo, a soma da primeira, penúltima e última linha produz o vector nulo. Assim, temos que $c = [1\ 0\ 0\ 0\ 0\ 1\ 1]$ é palavra deste código.

4.2.6 Código de Hamming (7,4) não sistemático

O código de Hamming (7,4) pode assumir a forma não sistemática [3, 6, 18]. Neste caso a matriz geradora é

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{26}$$

As palavras do código, organizadas na forma $\mathbf{c} = [b_0\ b_1\ m_0\ b_2\ m_1\ m_2\ m_3]$, constam da tabela 11. A matriz de teste de paridade e a sua transposta são definidas através de

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad \text{e} \quad \mathbf{H^T} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \tag{27}$$

respectivamente. Note-se que em alternativa a estas duas matrizes, podemos considerar outras versões destas, tais que se permutam as linhas (para $\mathbf{H}$) e as colunas (para $\mathbf{H^T}$), apresentadas em (27). A tabela 12 apresenta os síndromas e os respectivos padrões de 1 bit em erro. Pela análise da tabela, constata-se que o valor numérico do síndroma, entendido como número em binário natural a 3 bits, representa a posição do bit errado.

[6]Não é possível obter o vector nulo somando apenas duas linhas da matriz, uma vez que não existem duas linhas iguais.

Mensagem	Palavra de Código	Peso de Hamming
0000	0000000	0
0001	1101001	4
0010	0101010	3
0011	1000011	3
0100	1001100	3
0101	0100101	3
0110	1100110	4
0111	0001111	4
1000	1110000	3
1001	0011001	3
1010	1011010	4
1011	0110011	4
1100	0111100	4
1101	1010101	4
1110	0010110	3
1111	1111111	7

Tabela 11: Mensagens, palavras de código e peso de Hamming para o código Hamming (7,4) não sistemático.

Valor	Síndroma (s)	Padrão de erro (e)
0	000	0000000
1	001	1000000
2	010	0100000
3	011	0010000
4	100	0001000
5	101	0000100
6	110	0000010
7	111	0000001

Tabela 12: Tabela de síndromas e respectivos padrões de erro, para o código Hamming (7,4) não sistemático.

4.3 Análise comparativa de códigos

A tabela 13 apresenta uma análise comparativa dos códigos apresentados, relativamente ao *code rate* R e às capacidades de: detecção de erros; correcção de erros; detecção e correcção de erros. Para os códigos de repetição, verifica-se que com o aumento do número de bits redundantes, as capacidades de detecção e correcção de erros aumentam e a eficiência do código diminui. Os códigos de bit de paridade têm as mesmas capacidades de detecção e correcção, diferindo na eficiência. O mesmo se passa para os códigos de Hamming: dado que possuem sempre $d_{min} = 3$, tem-se que para estes códigos apenas muda a eficiência.

4.4 Modificações sobre códigos (n,k)*

Por vezes existem constrições que levam a modificações do comprimento das palavras de código a enviar por um determinado canal. A partir de um código (n,k) com características adequadas

Código	R	d_{min}	Detecção (l)	Correcção (t)	Det. e Corr. (l,t)
Repetição (2,1)	0.5	2	1	0	(-,-)
Repetição (3,1)	0.333	3	2	1	(-,-)
Repetição (4,1)	0.25	4	3	1	(2,1)
Repetição (5,1)	0.2	5	4	2	(3,1)
Paridade (3,2)	0.666	2	1	0	(-,-)
Paridade (8,7)	0.875	2	1	0	(-,-)
Hamming (7,4) r=3	0.571	3	2	1	(-,-)
Hamming (15,11) r=4	0.733	3	2	1	(-,-)
Hamming (31,26) r=5	0.838	3	2	1	(-,-)

Tabela 13: Análise comparativa de códigos.

de detecção e correcção de erros é adequado realizar modificações de forma a cumprir determinadas constrições do seu contexto de aplicação. Também é adequado partir de um código já estabelecido e realizar modificações nas suas dimensões em vez de criar um código de raíz. A tabela 14 apresenta as modificações a considerar, sobre um código de dimensões (n, k). As sub-secções seguintes exemplificam estas operações.

Operação	Dimensões	Características
Extensão	(n+1,k)	Adiciona um bit de paridade.
Redução	(n-1,k-1)	Remove um bit de mensagem.
Perfuração	(n-1,k)	Remove um bit de paridade.
Código dual	(n,q)	As matrizes G e H trocam de funcionalidade.

Tabela 14: Operações de modificação sobre um código de dimensões (n, k).

4.4.1 Extensão

A extensão de um código linear de bloco (n,k) consiste em acrescentar mais um bit de redundância, passando a ter a representação (n+1,k). Para os mesmos bits de mensagem existe mais um bit de controlo de paridade, o que aumenta a capacidade de controlo de erros, mas diminui o code rate (eficiência) do código. Em termos de espaço vectorial, o número de vectores que constituem a matriz geradora mantém-se. O comprimento das palavras aumenta de um. Se for efectuada a extensão de um código de Hamming (7,4) obtém-se um código (8,4). Um exemplo de extensão é acrescentar um bit de redundância que efectua a paridade de todos os bits anteriores que constituem a palavra. Seguindo este critério a matriz geradora do código de Hamming definido por (14) estendido é

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{28}$$

A tabela 15 apresenta as $2^4 = 16$ mensagens, as palavras do código e o respectivo peso de Hamming, para este código. Verifica-se que o código tem $R = \frac{4}{8} = 0.5$, d_{min}=4, tem l=3 e t=1. Este código consegue detecção e correcção simultâneas com $l = 2$ e $t = 1$.

Mensagem	Palavra de Código	Peso de Hamming
0000	00000000	0
0001	00011110	4
0010	00101011	4
0011	00110101	4
0100	01001101	4
0101	01010011	4
0110	01100110	4
0111	01111000	4
1000	10000111	4
1001	10011001	4
1010	10101100	4
1011	10110010	4
1100	11001010	4
1101	11010100	4
1110	11100001	4
1111	11111111	8

Tabela 15: Mensagens, palavras de código e peso de Hamming para o código Hamming (8,4).

4.4.2 Redução

A redução de um código (n,k) consiste em retirar um bit da mensagem, mantendo o número de bits redundantes, o que resulta num código (n-1,k-1). Para o código de Hamming (7,4), retirando o último bit de mensagem, obtém-se o código (6,3) cuja matriz geradora é

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \tag{29}$$

As palavras de código constam da tabela 16. O código tem $R = \frac{3}{6} = 0.5$, $\mathrm{d_{min}}$=3, l=2 e t=1.

Mensagem	Palavra de Código	Peso de Hamming
000	000000	0
001	001101	3
010	010110	3
011	011011	4
100	100011	3
101	101110	4
110	110101	4
111	111000	3

Tabela 16: Mensagens, palavras de código e peso de Hamming para o código (6,3).

Visto que se diminui o número de bits de mensagem mantendo o número de bits de redundância, a capacidade de controlo de erros do código reduzido será sempre igual ou superior à do código inicial.

4.4.3 Perfuração

A perfuração consiste na remoção de um bit de paridade. Por exemplo, o código (7,4) passa a (6,4) após perfuração de 1 bit de paridade. Normalmente a distância mínima do código perfurado é inferior à distância mínima do código original. Considerando a matriz geradora de código (7,4) definida por (14) e efectuando perfuração removendo o primeiro bit de paridade ficamos com

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \tag{30}$$

verificando-se que em termos matriciais, a perfuração consiste em remover uma coluna pertencente à sub-matriz geradora de paridade $\mathbf{P}$.

4.4.4 Código dual

Seja Cod um código (n,k) e Cod_2 outro código de dimensões (n,n-k). Os códigos Cod e Cod_2 dizem-se duais, caso possuam as seguintes características [18]:

- a matriz geradora de Cod é a matriz de teste de paridade de Cod_2;
- a matriz de teste de paridade de Cod é a matriz geradora de Cod_2.

As palavras de ambos os códigos são ortogonais entre si. Analisando as palavras de código enquanto elementos de sub-espaço vectorial, verifica-se que para qualquer código, as matrizes $\mathbf{G}$ e $\mathbf{H}$ geram espaços vectoriais ortogonais e ambos contêm o vector nulo.

Através de (19) mostra-se que todos os produtos internos entre as linhas de $\mathbf{G}$ e as colunas de $\mathbf{H^T}$ são nulos. Como as colunas de $\mathbf{H^T}$ são as linhas de $\mathbf{H}$, então as linhas de $\mathbf{G}$ são ortogonais a todas as linhas de $\mathbf{H}$. Considerando um código de Hamming (7,4) com matrizes geradora e de teste de paridade definidas por

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad \text{e} \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \tag{31}$$

respectivamente, verifica-se que H gera um código (7,3), uma vez que tem dimensões 3×7, e é constituída por vectores linearmente independentes.

Em termos de espaços vectoriais podemos verificar que temos um espaço vectorial V de dimensão 7. Este espaço vectorial é decomposto em dois sub-espaços vectoriais ortogonais. As palavras do código (7,4) pertencem ao sub-espaço de dimensão 4, enquanto que as palavras do código (7,3) pertencem ao sub-espaço de dimensão 3. A dimensão de um espaço define-se como o número máximo de vectores linearmente independentes existentes nesse espaço [18]. A figura 20 ilustra esta decomposição.

Outras formas de modificação de códigos lineares estão em [18].

5 Códigos lineares de bloco cíclicos

Os códigos lineares de bloco cíclicos constituem uma sub-classe dos códigos lineares de bloco[7]. Para além de possuirem as características enunciadas acima para os códigos lineares de bloco,

[7]Um código cíclico não é necessariamente linear. No entanto, caso não seja linear, a sua estrutura é menos interessante do ponto de vista prático [18].

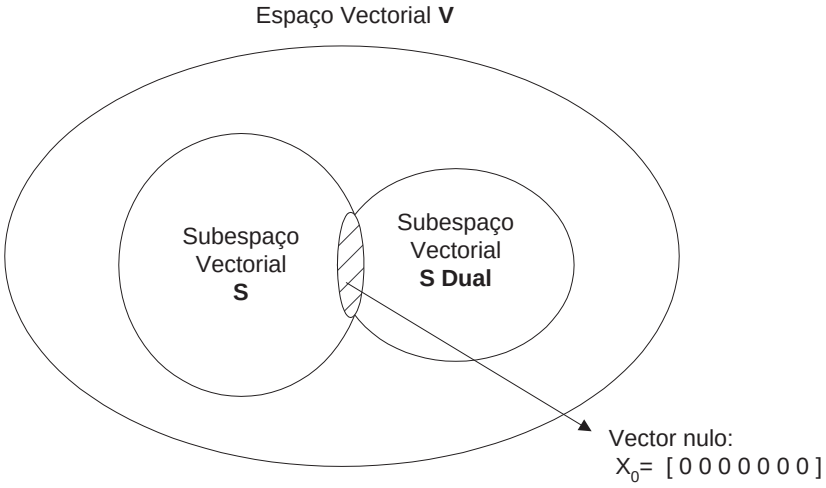


Figura 20: Decomposição dum espaço vectorial em dois sub-espaços (duais). O vector nulo pertence simultaneamente ao sub-espaço e ao sub-espaço dual.

verificam ainda a propriedade de que uma rotação cíclica de qualquer ordem sobre qualquer palavra do código produz outra palavra do código. Por exemplo, aplicando a rotação cíclica de ordem 1 para a esquerda sobre a palavra

$$\mathbf{c} = [c_{n-1}\ c_{n-2}\ c_{n-3}\ \ldots\ c_0],$$

obtém-se

$$\mathbf{c}' = [c_{n-2}\ c_{n-3} \ldots\ c_0\ c_{n-1}].$$

Aplicando outra rotação para a esquerda, a palavra de código fica

$$\mathbf{c}'' = [c_{n-3}\ \ldots\ c_0\ c_{n-1}\ c_{n-2}].$$

5.1 Palavras de código como polinómios

O estudo dos códigos cíclicos é facilitado, caso as palavras de código sejam analisadas como polinómios. A palavra de código

$$\mathbf{c} = [c_{n-1}\ c_{n-2}\ \ldots\ c_1\ \ c_0].$$

escrita na forma polinomial fica

$$c(X) = c_{n-1}X^{n-1} + c_{n-2}X^{n-2} + \ldots + c_1X + c_0, \tag{32}$$

com $c_i\ \in \{0, 1\}$.

5.1.1 Operações sobre polinómios

De forma a facilitar a restante exposição sobre códigos cíclicos, esta sub-secção contém uma revisão das operações sobre polinómios. As operações a considerar são multiplicação por X e X^k, adição e rotação. Considerem-se $c(X)$ e $d(X)$ dois polinómios correspondentes a palavras de código.

- **Multiplicação por** X - sobre o polinómio $c(X)$, definido por (32) a multiplicação por X resulta em $c(X)X = c_{n-1}X^n + c_{n-2}X^{n-1} + \ldots + c_1X^2 + c_0X$;

- **Multiplicação por** X^k - nesta situação obtemos $c(X)X^k = c_{n-1}X^{n-1+k} + c_{n-2}X^{n-2+k} + \ldots + c_1X^{1+k} + c_0X^k$;
- **Adição** - a adição de dois polinómios $c(X)$ e $d(X)$, do mesmo grau, em aritmética de módulo 2, resulta em

  $$c(X) \oplus d(X) = (c_{n-1} \oplus d_{n-1})X^{n-1} + (c_{n-2} \oplus d_{n-2})X^{n-2} + \ldots + (c_1 \oplus d_1)X + (c_0 \oplus d_0).$$

  Sempre que se verifique $c(X) = d(X)$, a soma dá o polinómio (vector) nulo.
- **Rotação** - a rotação cíclica para a esquerda do polinómio $c(X)$ é dada por

  $$c'(X) = c_{n-2}X^{n-1} + \ldots + c_1X^2 + c_0X + c_{n-1}.$$

Note-se que uma rotação para a direita pode ser obtida à custa de $n-1$ rotações para a esquerda. Verifica-se que a rotação para a esquerda é obtida, analiticamente pela operação

$$c'(X) = c(X)X + c_{n-1}(X^n + 1), \tag{33}$$

a qual consiste na multiplicação do polinómio por X, para deslocar uma posição para a esquerda

$$c(X)X = c_{n-1}X^n + c_{n-2}X^{n-1} + \ldots + c_1X^2 + c_0X,$$

e em seguida, para anular o termo $c_{n-1}X^n$ e obter c_{n-1}, é necessário somar estes dois termos à expressão anterior, obtendo-se

$$c(X)X + c_{n-1}X^n + c_{n-1} = \underbrace{c_{n-1}X^n + c_{n-1}X^n}_{0} + c_{n-2}X^{n-1} + \ldots + c_1X^2 + c_0X + c_{n-1}.$$

Verifica-se finalmente que

$$\begin{aligned} c'(X) &= c(X)X + c_{n-1}(X^n + 1) \\ &= c_{n-2}X^{n-1} + \ldots + c_1X^2 + c_0X + c_{n-1}. \end{aligned} \tag{34}$$

5.2 Polinómio gerador

Todas as palavras de um código cíclico (n,k) são obtidas a partir do polinómio gerador do código $g(X)$, de coeficientes binários. Cada palavra de código $c_i(X)$ é escrita na forma

$$c_i(X) = f_i(X)g(X), \tag{35}$$

em que $f_i(X)$ é outro polinómio de grau $k-1$, único para cada palavra do código. Contudo, não é forçoso que $f_i(X)$ seja o polinómio mensagem, ou seja, os bits de mensagem na forma polinomial. Para $g(X)$ ser polinómio gerador de um código de bloco cíclico (n,k), com $q = n-k$ bits redundantes, deve ter as seguintes propriedades [7, 10, 18]:

1. grau q;
2. ser factor de $X^n + 1$.

A primeira propriedade implica que $g(X)$ é da forma $g(X) = X^q + \ldots$. Apenas assim é possível obter em $c_i(X)$ um polinómio de grau $n-1$ uma vez que $n-1 = (k-1)+q = k+q-1$.

A segunda propriedade (factorização) indica que o resto da divisão de X^n+1 por $g(X)$ deve ser zero

$$\text{rem}\left(\frac{X^n+1}{g(X)}\right) = 0, \tag{36}$$

em que *rem* representa o resto da divisão dos dois polinómios. De forma equivalente temos

$$\frac{X^n+1}{g(X)} = h(X) + \frac{0}{g(X)} \qquad (=) \qquad X^n+1 = g(X)h(X), \tag{37}$$

de acordo com a forma geral da divisão de polinómios

$$\frac{\alpha(x)}{\beta(x)} = q(x) + \frac{r(x)}{\beta(x)}. \tag{38}$$

Esta condição implica que o termo de grau zero de $g(X)$ vale 1, concluindo-se que o polinómio gerador é da forma $g(X) = X^q + \ldots + 1$. O polinómio $h(X)$, apresentado em (37), designa-se por polinómio de teste de paridade.

5.2.1 Relação com a matriz geradora

As palavras dos códigos de bloco lineares são obtidas a partir da combinação linear das linhas da matriz geradora **G**. As palavras dos códigos de bloco lineares cíclicos, são estabelecidas a partir do polinómio gerador do código. Por outro lado, um código cíclico, dado que é linear, também possui matriz geradora. Esta pode ser obtida a partir do polinómio gerador, efectuando sucessivas rotações sobre este. Por exemplo, seja o polinómio gerador $g(X) = X^q + g_{q-1}X^{q-1} + g_{q-2}X^{q-2} + \ldots + g_1X^1 + 1$. A matriz geradora do código (n,k), constituída por k palavras de código de dimensão n, pode ser obtida através de

$$\mathbf{G} = \begin{bmatrix} 1 & g_{q-1} & g_{q-2} & \ldots & g_1 & 1 & \ldots & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & g_{q-1} & g_{q-2} & \ldots & g_1 & 1 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & g_{q-1} & g_{q-2} & \ldots & g_1 & 1 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & \end{bmatrix}. \tag{39}$$

O polinómio $g(X)$, descreve o código estabelecendo-se a matriz geradora a partir deste polinómio. A partir do polinómio de teste de paridade $h(X)$, obtém-se a matriz de teste de paridade **H**, tal que $\mathbf{GH^T} = \mathbf{0}$.

A matriz obtida em (39) não assume normalmente a forma sistemática. Para o conseguir, é necessário realizar as substituições convenientes até figurar a matriz identidade e a sub-matriz geradora de paridade. Estas substituições devem assegurar que as palavras da matriz geradora obtida são linearmente independentes. As novas palavras a introduzir em substituição das previamente existentes são obtidas através da soma e remoção destas, tal como apresentado na sub-secção 5.3.3.

5.3 Códigos cíclicos sistemáticos

À semelhança dos códigos lineares de bloco, os códigos cíclicos também se apresentam na forma sistemática ou não sistemática. A condição de código sistemático em que as palavras

de código estão organizadas em blocos de mensagem e de bits de paridade, assumindo a forma $\mathbf{c} = [m_0\ m_1\ \ldots\ m_{k-1}\ b_0\ b_1\ \ldots\ b_{q-1}]$ que em termos polinomiais fica

$$c_i(X) = m_i(X)X^q + b(X), \tag{40}$$

em que $b(X)$ é o polinómio que representa os bits de paridade. A multiplicação por X^q desloca os bits de mensagem, para a esquerda em q posições. Em (35) e (40), definem-se as condições de cíclico e cíclico e sistemático, respectivamente. Desta forma, para o código ser cíclico e sistemático deve verificar simultaneamente as duas condições

$$c_i(X) = f_i(X)g(X) = m_i(X)X^q + b(X), \tag{41}$$

para todas as palavras de código $c_i(X)$. A conjunção destas condições obriga a que os bits de paridade, dados pelo polinómio $b(X)$ sejam calculados de determinada forma. Demonstra-se de seguida a forma de cálculo destes bits de paridade. Dividindo ambos os termos de (41) por $g(X)$ obtém-se

$$\begin{aligned} \frac{f_i(X)g(X)}{g(X)} &= \frac{m_i(X)X^q + b(X)}{g(X)} \\ (=)\ f_i(X) &= \frac{m_i(X)X^q}{g(X)} + \frac{b(X)}{g(X)} \\ (=)\ f_i(X) + \frac{m_i(X)X^q}{g(X)} &= \frac{b(X)}{g(X)} \end{aligned} \tag{42}$$

De acordo com a expressão geral da divisão de polinómios (38), temos que (42) assume a forma

$$\frac{m_i(X)X^q}{g(X)} = f_i(X) + \frac{b(X)}{g(X)}. \tag{43}$$

Conclui-se assim que o polinómio $b(X)$ resulta do resto da divisão de $m_i(X)X^q$ por $g(X)$

$$b(X) = \text{rem}\left(\frac{m_i(X)X^q}{g(X)}\right), \tag{44}$$

sendo normalmente designado por CRC (*Cyclic Redundancy Check*) [7, 10, 18]. A palavra de código linear de bloco cíclico e sistemático é então expressa na forma

$$c_i(X) = m_i(X)X^q + b(X) = m_i(X)X^q + \text{rem}\left(\frac{m_i(X)X^q}{g(X)}\right). \tag{45}$$

5.3.1 Codificação

O CRC consiste no resto da divisão da mensagem deslocada de q bits para a esquerda pelo polinómio gerador. Tendo em conta (45), podemos assim sumarizar a obtenção do CRC:

1. multiplicar o polinómio mensagem $m_i(X)$ por X^q, obtendo $m_i(X)X^q$;
2. dividir $m_i(X)X^q$ pelo polinómio gerador $g(X)$ e obter o resto dessa divisão $b(X)$;
3. somar $b(X)$ a $m_i(X)X^q$ obtendo $c_i(X)$.

A figura 21 ilustra a disposição dos bits de mensagem e de paridade, num código cíclico sistemático.

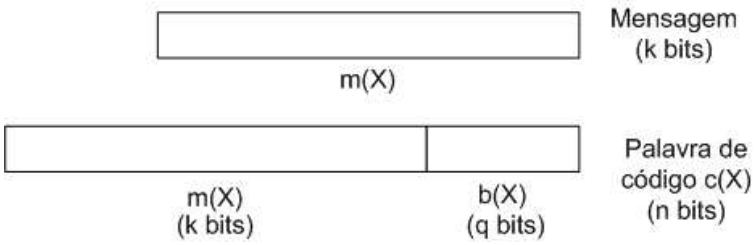


Figura 21: Disposição dos bits de mensagem e de paridade, num código cíclico sistemático.

5.3.2 Descodificação

Qualquer de palavra de código é escrita na forma

$$c_i(X) = f_i(X)g(X).$$

Desta forma, todas as palavras de código são factor do polinómio gerador e verificam

$$\text{rem}\left(\frac{c_i(X)}{g(X)}\right) = 0,$$

ou seja, obtém-se resto nulo na divisão da palavra de código pelo polinómio gerador. Caso a palavra recebida tenha erros, então pode ser escrita na forma

$$y_i(X) = c_i(X) + e(X),$$

em que $e(X)$ é o polinómio que representa o padrão de erro. Neste caso, a divisão da palavra de código pelo polinómio gerador é dada por

$$\frac{c_i(X) + e(X)}{g(X)} = \frac{f_i(X)g(X) + e(X)}{g(X)}.$$

O resto desta divisão fica

$$s(X) = \text{rem}\left(\frac{f_i(X)g(X) + e(X)}{g(X)}\right) = \text{rem}\left(\frac{e(X)}{g(X)}\right).$$

Assim temos que $s(X)$ é nulo quando $e(X)$ é nulo, ou seja, obtemos síndroma nulo sempre que não exista erro detectado. O polinómio $s(X)$ designa-se de síndroma e depende apenas do padrão de erro $e(X)$.

5.3.3 Exemplos de polinómios geradores e respectivos códigos

O polinómio $g(X) = X + 1$ gera um código cíclico (3,2). A figura 22 mostra a divisão $\frac{X^3+1}{X+1}$, a qual tem resto zero (condição necessária e suficiente para ser polinómio gerador do código). A tabela 17 apresenta as palavras de código, através das quais se verifica que qualquer rotação sobre qualquer palavra produz outra palavra de código. Verifica-se que se trata do código de bloco (3,2) de bit de paridade par, apresentado na secção 3.2.4. Para este código, podemos estabelecer, por exemplo, as matrizes geradoras

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \qquad \mathbf{G} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \qquad \text{e} \qquad \mathbf{G} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{46}$$

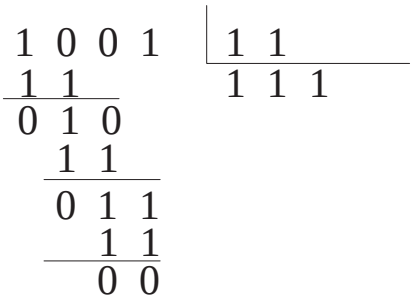


Figura 22: Divisão de polinómios $\frac{X^3+1}{X+1}$; polinómio gerador do código (3,2).

Palavras do código (3,2)
000
011
110
101

Tabela 17: Palavras de código do código cíclico (3,2).

A partir da divisão apresentada na figura 22, concluímos que o polinómio $X^2 + X + 1$ também é factor de $X^3 + 1$ e como tal gera um código cíclico (3,1), cujas palavras são [0 0 0] e [1 1 1]. Este é o código de repetição (3,1), tratado na secção 3.2.3.

$$\frac{X^3+1}{X+1} = X^2 + X + 1 \qquad (=) \qquad X^3 + 1 = (X+1)(X^2+X+1).$$

O polinómio $g(X) = X^3 + X + 1$ gera um código cíclico. A figura 23 ilustra a divisão $\frac{X^7+1}{X^3+X+1}$. Verifica-se que a divisão apresenta resto nulo e como tal garante-se que $g(X)$ gera um

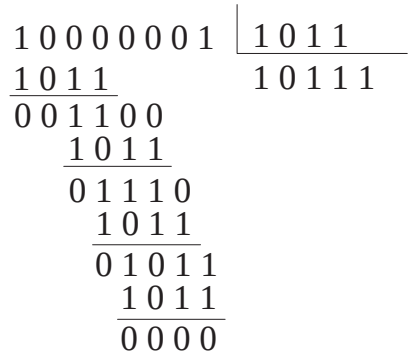


Figura 23: Divisão de polinómios $\frac{X^7+1}{X^3+X+1}$; polinómio gerador do código (7,4).

código cíclico (7,4). A partir do polinómio gerador, podemos estabelecer a matriz geradora do código

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \tag{47}$$

na forma não sistemática, efectuando sucessivas rotações para a direita sobre este, obtendo palavras de código linearmente independentes. Para obter a matriz geradora na forma sistemática e dado que o código é linear efectuam-se adições de palavras de código, obtendo a forma desejada para a matriz, garantindo que as palavras são linearmente independentes. A matriz

$$\mathbf{G_{sist}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \tag{48}$$

define a forma sistemática do código. A primeira linha de $\mathbf{G_{sist}}$ é obtida pela soma da 1ª, 3ª e 4ª linhas de $\mathbf{G}$, enquanto que a segunda linha consiste na soma da 2ª e 4ª linhas $\mathbf{G}$. Pela análise da figura 4, verifica-se que

$$\frac{X^7+1}{X^3+X+1} = X^4+X^2+X+1 \qquad (=) \qquad X^7+1 = (X^3+X+1)(X^4+X^2+X+1),$$

ou seja, o polinómio X^4+X^2+X+1 gera um código cíclico (7,3).

5.3.4 Factorização de X^n+1 e ploinómios geradores standard

O polinómio gerador de um código (n,k) é factor de X^n+1 e como tal a factorização de X^n+1 em polinómios de coeficientes binários assume especial importância, para diferentes valores de n. A tabela 18 apresenta exemplos dessa factorização. Através da consulta da tabela obtêm-se

$\mathbf{X^n+1}$	**Produto de polinómios**
X^3+1	$X^3+1=(X+1)(X^2+X+1)$
X^5+1	$X^5+1=(X+1)(X^4+X^3+X^2+X+1)$
X^7+1	$X^7+1=(X+1)(X^3+X^2+1)(X^3+X+1)$
X^9+1	$X^9+1=(X+1)(X^2+X+1)(X^6+X^3+1)$
$X^{11}+1$	$X^{11}+1=(X+1)(X^{10}+X^9+X^8+X^7+X^6+\ldots$
	$\ldots+X^5+X^4+X^2+X+1$

Tabela 18: Factorização de X^n+1 em polinómios de coeficientes binários.

polinómios geradores de diferentes graus. Por exemplo, os polinómios X^3+X^2+1 e X^3+X+1 geram códigos com $q=3$ bits redundantes. Dado que são factor de X^7+1 ambos geram códigos $(7,7-3)=(7,4)$. O polinómio $(X+1)$ também gera um código cíclico com 1 bit redundante - código (7,6), por exemplo. A factorização assegura que o polinómio gera um código cíclico; no entanto, não assegura que este seja um *bom* código cíclico, no que diz respeito à sua distância mínima e consequentemente em relação às capacidades de detecção e correcção de erros.

A tabela 19 apresenta polinómios geradores standard [18]. O número de bits redundantes do código corresponde ao grau do polinómio. Os polinómios CRC16 code e CRC-CCITT são utilizados em protocolos de transferência de dados em WAN (*Wide Area Network*) enquanto que o polinómio CRC32 code é utilizado em LAN (*Local Area Network*) [5, 14], originando conjuntos de bits redundantes, normalmente designados de FCS (*Frame Check Sequence*).

Código CRC	Polinómio gerador
CRC4 code	$g(X) = X^4 + X^3 + X^2 + X + 1$
CRC7 code	$g(X) = X^7 + X^6 + X^4 + 1$
CRC12 code	$g(X) = X^{12} + X^{11} + X^3 + X^2 + X + 1$
CRC16 code	$g(X) = X^{16} + X^{15} + X^2 + 1$
CRC-CCITT	$g(X) = X^{16} + X^{12} + X^5 + 1$
CRC32 code	$g(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + \ldots$
	$\ldots + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

Tabela 19: Alguns polinómios geradores standard. CCITT significa *Comité Consultatif International Télégraphique et Téléphonique*.

5.3.5 Códigos de Hamming cíclicos e não cíclicos

Os códigos de Hamming[8] [2, 3, 6, 7] podem assumir a forma cíclica, escolhendo as equações de paridade de forma criteriosa. Os códigos de Hamming são definidos por um parâmetro inteiro r (≥ 2) tal que: $(n, k) = (2^r - 1, 2^r - 1 - r)$; com $r = 3$ tem-se k=4 e n=7. O código de Hamming (7,4) com as equações de paridade

$$\begin{aligned} b_0 &= m_1 \oplus m_2 \oplus m_3, \\ b_1 &= m_0 \oplus m_1 \oplus m_3, \\ b_2 &= m_0 \oplus m_2 \oplus m_3, \end{aligned} \tag{49}$$

definido pela matriz geradora

$$\mathbf{G} = \left[\begin{array}{cccc} m_0 & m_1 & m_2 & m_3 \end{array} \right] \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right], \tag{50}$$

é não cíclico. Por exemplo, a rotação para a direita, da segunda linha de $\mathbf{G}$, não produz a terceira linha de $\mathbf{G}$. Os códigos de Hamming podem assumir forma cíclica. O polinómio

$$g(X) = X^3 + X + 1$$

de grau q=3 gera um código de Hamming $(7, 4)$ cíclico. Na figura 23 e na tabela 18 verifica-se que este polinómio é factor de $X^7 + 1$. A matriz geradora na forma sistemática, obtida a partir deste polinómio, apresentada em (48) gera um código de Hamming (7,4) cíclico. A tabela 20 apresenta todas as palavras de código.

O polinómio $g(X) = X^3 + X^2 + 1$ também gera um código cíclico de Hamming (7,4) [18]. Procedendo de forma idêntica à anterior, obtêm-se todas as palavras de código, as quais se apresentam na tabela 21.

5.4 Capacidades de detecção e correcção

Dado que o código cíclico é um código linear de bloco, as capacidades de detecção e correcção de erros dos códigos, em função da distância mínima, mantêm-se relativamente aos códigos

[8]Richard W. Hamming (1915-1998) `http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Hamming.html`

Palavra de Código	Peso de Hamming
0000 000	0
0001 011	3
0010 110	3
0011 101	4
0100 111	4
0101 100	3
0110 001	3
0111 010	4
1000 101	3
1001 110	4
1010 011	4
1011 000	3
1100 010	3
1101 001	4
1110 100	4
1111 111	7

Tabela 20: Palavras de código e respectivo peso de Hamming para o código Hamming (7,4) cíclico sistemático, gerado por $g(X) = X^3 + X + 1$.

Palavra de Código	Peso de Hamming
0000 000	0
0001 101	3
0010 111	4
0011 010	3
0100 011	3
0101 110	4
0110 100	3
0111 001	4
1000 110	3
1001 011	4
1010 001	3
1011 100	4
1100 101	4
1101 000	3
1110 010	4
1111 111	7

Tabela 21: Palavras de código e respectivo peso de Hamming para o código Hamming (7,4) cíclico sistemático, gerado por $g(X) = X^3 + X^2 + 1$.

lineares de bloco. No entanto, dada a sua estrutura, os códigos cíclicos são adequados para a detecção de erros, nomeadamente de *burst* de erros, na situação em que $n \gg 1$. Um *burst* de comprimento B, numa palavra de n bits, é uma sequência contígua de B bits, na qual o primeiro e o último bit são recebidos em erro. Os códigos cíclicos (n,k) detectam [7]:

- todos os *burst* de comprimento igual ou inferior a $n-k$;
- uma fracção dos *burst* de comprimento $n-k+1$; esta fracção é $1-2^{-(n-k-1)}$;
- uma fracção dos *burst* de comprimento maior que $n-k+1$; esta fracção é $1-2^{-(n-k)}$;
- todas as combinações de $\mathrm{d_{min}}-1$ ou menos erros;
- todos os padrões de erro com número ímpar de erros, se o polinómio gerador possuir um número par de coeficientes não nulos.

Por exemplo, considerando o polinómio **CRC7 code** $g(X)=X^7+X^6+X^4+1$, apresentado na tabela 19, verifica-se que este detecta:

- todos os *burst* de comprimento igual ou inferior a 7;
- $1-2^{-(7-1)}=98.44$ % dos *burst* de comprimento 8;
- $1-2^{-7}=99.22$ % dos *burst* de comprimento maior que 8;
- todos os padrões de erro com número ímpar de erros.

Dado que um código linear de bloco cíclico é também um código linear de bloco, tem-se que a distância mínima do código obtido está relacionada com o mínimo peso de Hamming, tal como indicado por(11).

5.5 Codificação e descodificação - realização em *hardware**

Uma vantagem dos códigos cíclicos em relação aos códigos de bloco, é a realização do codificador e do descodificador por *hardware*, utilizando *flip-flops*, organizados na estrutura *shift-register*. Esta realização não necessita de cálculo matricial para obter o síndroma, sendo eficiente relativamente à memória ocupada e ao tempo de execução.

5.5.1 Codificação

A figura 24 mostra o codificador do código de Hamming (7,4) cíclico sistemático, cujo polinómio gerador é $g(X)=X^3+X+1$. O codificador funciona da seguinte forma:

1. Fechar a *gate* e colocar o *switch* na posição 1;
2. Ler os k bits da mensagem (*shift-in*) (k clocks);
3. Abrir a *gate* e colocar o *switch* na posição 2;
4. Enviar para a saída o conteúdo do *shift register* (*shift-out*) (q clocks).

Note-se que a estrutura do codificador é definida pelo polinómio gerador do código; os termos X^3, X e 1 definem ligações, enquanto que o coeficiente da potência X^2 ao ser zero, não estabelece ligação entre b_1 e b_2. A estrutura é ainda simplificada pelo facto do código ser sistemático, separando os bits de mensagem dos bits de paridade. O codificador é descrito pelas expressões *booleanas*

$$\begin{aligned} b_0 &= \mathrm{input} \oplus b_2^*, \\ b_1 &= b_0^* \oplus b_2^* \oplus \mathrm{input}, \\ b_2 &= b_1^*, \end{aligned} \tag{51}$$

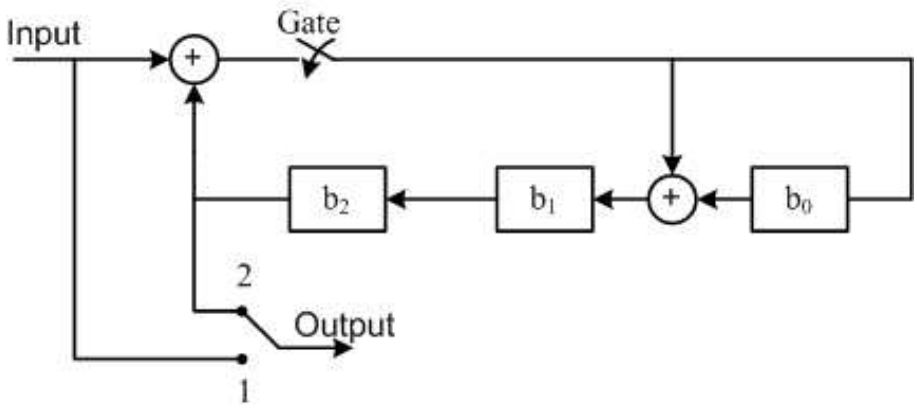


Figura 24: Codificador de código cíclico sistemático Hamming (7,4) com polinómio gerador $g(X) = X^3 + X + 1$; realização do codificador em *hardware*.

em que b_i^* representa o estado anterior do *flip-flop* b_i. A tabela 22 apresenta a evolução do *shift-register* e os bits na saída quando a mensagem é $m(X) = X^3 + X^2$, ou na forma de vector $\mathbf{m} = [1\ 1\ 0\ 0]$. A palavra de código obtida é $\mathbf{c} = [1\ 1\ 0\ 0\ 0\ 1\ 0]$, tal como consta da tabela 20.

Bit mensagem m_i	$\mathbf{b_2}$	$\mathbf{b_1}$	$\mathbf{b_0}$	**Bit na saída**
	0	0	0	
1	0	1	1	1
1	1	0	1	1
0	0	0	1	0
0	0	1	0	0
	1	0	0	0
	0	0	0	1
	0	0	0	0

Tabela 22: Cálculo de palavra de código cíclico Hamming (7,4) com polinómio gerador $g(X) = X^3 + X + 1$ e mensagem $m(X) = X^3 + X^2$; realização do codificador em *hardware*.

A tabela 23 apresenta a codificação da mensagem $m(X) = 1$, ou seja, $\mathbf{m} = [0\ 0\ 0\ 1]$. A palavra de código obtida é $\mathbf{c} = [0\ 0\ 0\ 1\ 0\ 1\ 1]$ (ver tabela 20).

Bit mensagem m_i	$\mathbf{b_2}$	$\mathbf{b_1}$	$\mathbf{b_0}$	**Bit na saída**
	0	0	0	
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	1	1	1
	1	1	0	0
	1	0	0	1
	0	0	0	1

Tabela 23: Cálculo de palavra de código cíclico Hamming (7,4) com polinómio gerador $g(X) = X^3 + X + 1$ e mensagem $m(X) = 1$; realização do codificador em *hardware*.

5.5.2 Descodificação

A descodificação baseia-se na divisão da palavra de código pelo polinómio gerador, obtendo assim o síndroma correspondente. O descodificador pode então ser visto como uma calculadora de síndromas. A figura 25 apresenta a calculadora de síndromas para o código cujo codificador se apresentou na figura 24 (com polinómio gerador $g(X) = X^3 + X + 1$). Este descodificador é descrito pelas expressões *booleanas*

$$\begin{aligned} s_0 &= \text{input} \oplus s_2^*, \\ s_1 &= s_0^* \oplus s_2^*, \\ s_2 &= s_1^*, \end{aligned} \tag{52}$$

em que s_i^* representa o estado anterior do *flip-flop* designado por s_i. O funcionamento do

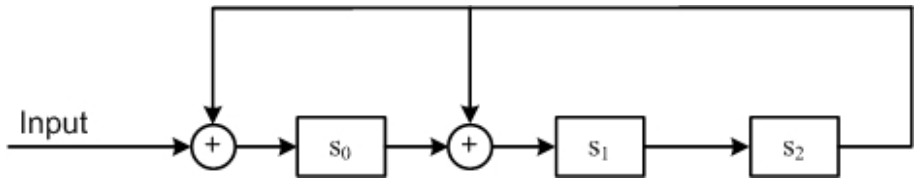


Figura 25: Cálculo de síndroma para o código cíclico Hamming (7,4) com polinómio gerador $g(X) = X^3 + X + 1$; realização do descodificador em *hardware*.

descodificador é o seguinte:

1. Ler os n bits da palavra de código (*shift-in*), pela mesma ordem com que foram enviados pelo codificador;
2. O síndroma corresponde ao conteúdo do *shift register*.

A tabela 24 apresenta a evolução do *shift-register* ao longo do processo de descodificação (cálculo do síndroma) da palavra de código $c(X) = X^6+X^5+X$, ou na forma de vector $\mathbf{c} = [1\ 1\ 0\ 0\ 0\ 1\ 0]$. A última linha da tabela contém o síndroma. Dado que a palavra pertence ao código, o síndroma obtido é nulo. A tabela 25 apresenta o cálculo do síndroma na situação em que a palavra de

Input	$\mathbf{s_0}$	$\mathbf{s_1}$	$\mathbf{s_2}$
	0	0	0
1	1	0	0
1	1	1	0
0	0	1	1
0	1	1	1
0	1	0	1
1	0	0	0
0	**0**	**0**	**0**

Tabela 24: Cálculo do síndroma para palavra de código cíclico Hamming (7,4) com polinómio gerador $g(X) = X^3 + X + 1$; ausência de erros; realização do descodificador em *hardware*.

código tem o último bit em erro $\mathbf{c} = [1\ 1\ 0\ 0\ 0\ 1\ 1]$. O síndroma obtido é $\mathbf{s} = [s_0\ s_1\ s_2] = [1\ 0\ 0]$. O cálculo do síndroma para a palavra $\mathbf{c} = [0\ 0\ 1\ 0\ 0\ 0\ 0]$ é apresentado na tabela 26. Esta palavra é obtida por troca do terceiro bit sobre o vector nulo[9], portanto tem o terceiro bit errado. O síndroma obtido é $\mathbf{s} = [s_0\ s_1\ s_2] = [0\ 1\ 1]$.

[9]Palavra de código, dado que o código é linear.

Input	$\mathbf{s_0}$	$\mathbf{s_1}$	$\mathbf{s_2}$
	0	0	0
1	1	0	0
1	1	1	0
0	0	1	1
0	1	1	1
0	1	0	1
1	0	0	0
1	**1**	**0**	**0**

Tabela 25: Cálculo do síndroma para palavra não pertencente ao código cíclico Hamming (7,4) com polinómio gerador $g(X) = X^3 + X + 1$; último bit em erro; realização do descodificador em *hardware*.

Input	$\mathbf{s_0}$	$\mathbf{s_1}$	$\mathbf{s_2}$
	0	0	0
0	0	0	0
0	0	0	0
1	1	0	0
0	0	1	0
0	0	0	1
0	1	1	0
0	**0**	**1**	**1**

Tabela 26: Cálculo do síndroma para palavra não pertencente ao código cíclico Hamming (7,4) com polinómio gerador $g(X) = X^3 + X + 1$; terceiro bit em erro; realização do descodificador em *hardware*.

A matriz geradora deste código está apresentada em (48). A matriz de teste de paridade transposta é

$$\mathbf{H^T} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

A tabela 27 apresenta os síndromas e os respectivos padrões de erro, para este código. Note-se que os síndromas estão apresentados na forma $\mathbf{s} = [s_2\ s_1\ s_0]$. Analisando a tabela, verifica-se que os síndromas obtidos nos cálculos apresentados nas tabelas 25 e 26, correspondem aos padrões de erro introduzidos sobre as mesmas.

6 Utilização em MATLAB

Nesta secção apresentam-se exemplos de utilização do MATLAB, nomeadamente algumas funcionalidades da Communications Toolbox, para codificação, descodificação, detecção e correcção

Síndromas $= [s_2\ s_1\ s_0]$	**Padrão de erro (e)**
000	0000000
101	1000000
111	0100000
110	**0010000**
011	0001000
100	0000100
010	0000010
001	**0000001**

Tabela 27: Tabela de síndromas e respectivos padrões de erro, para o código Hamming (7,4), definido por (48).

de erros. São explorados alguns aspectos dos códigos de bloco linear cíclicos e não cíclicos.

6.1 Códigos lineares de bloco - codificação e descodificação

O troço de código seguinte ilustra o estabelecimento da matriz geradora e de teste de paridade para um código de Hamming (7,4), utilizando a função `hammgen`. O parâmetro de entrada é o factor r de desenho do código, tal que $(n, k) = (2^r - 1, 2^r - 1 - r)$. A função `hammgen` retorna também os valores de n e k.

```
>> r=3; [H,G,n,k] = hammgen(r)
H = 1  0  0  1  0  1  1
    0  1  0  1  1  1  0
    0  0  1  0  1  1  1

G = 1  1  0  1  0  0  0
    0  1  1  0  1  0  0
    1  1  1  0  0  1  0
    1  0  1  0  0  0  1

n = 7

k = 4
```

Note-se que a matriz geradora está na forma sistemática $\mathbf{G} = [P \mid I_4]$. A sub-matriz geradora de paridade $\mathbf{P}$ difere da apresentada em (14); trata-se de outro código de Hamming.

Exemplifica-se agora a obtenção de uma palavra de código, de duas formas distintas: realizando a multiplicação do vector mensagem pela matriz geradora; somando, de forma modular $\oplus$, tal como estabelecido em (6), as duas linhas correspondentes da matriz geradora.

```
>> msg = [0 0 1 1];
>> c = mod(msg*G,2)
c = 0  1  0  0  0  1  1

>> c = mod(G(3,:) + G(4,:),2)
c = 0  1  0  0  0  1  1
```

Sobre a palavra de código, introduz-se um erro, trocando o segundo bit desta, através da soma com um padrão de erro. Calcula-se o síndroma e verifica-se que este é a segunda linha de $\mathbf{H^T}$, correspondendo ao padrão de erro somado à palavra.

```
>> e = [0 1 0 0 0 0 0];  y = mod(c + e,2)
y = 0  0  0  0  0  1  1

>> s=mod(y*H',2)
s = 0  1  0

>> H'
ans =
     1  0  0
     0  1  0
     0  0  1
     1  1  0
     0  1  1
     1  1  1
     1  0  1
```

A correcção, a partir da tabela de síndromas/padrões de erro, realiza-se da forma indicada de seguida. Define-se a tabela de padrões de erro, incluindo o padrão de erro nulo. Sabendo que o síndroma obtido corresponde à segunda linha de $\mathbf{H^T}$, é necessário indexar à tabela padrões de erro na posição respectiva (3ª linha, devido à inclusão do padrão de erro nulo na tabela). Após obter o padrão de erro, basta somá-lo à palavra recebida para obter a palavra original.

```
>> Err = [0 0 0 0 0 0 0; 1 0 0 0 0 0 0; 0 1 0 0 0 0 0; 0 0 1 0 0 0 0;
          0 0 0 1 0 0 0; 0 0 0 0 1 0 0; 0 0 0 0 0 1 0; 0 0 0 0 0 0 1]
Err =
     0     0     0     0     0     0     0
     1     0     0     0     0     0     0
     0     1     0     0     0     0     0
     0     0     1     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     1     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     1

>> err_pat = Err(3,:)
err_pat =
     0     1     0     0     0     0     0

>> c_est= mod( y + err_pat, 2)
c_est =
     0     1     0     0     0     1     1
```

Passando a analisar o código de Hamming definido pelo parâmetro $r = 4$, temos

```
>> r=4; [H,G,n,k] = hammgen(r)
H =  1  0  0  0  1  0  0  1  1  0  1  0  1  1  1
     0  1  0  0  1  1  0  1  0  1  1  1  1  0  0
     0  0  1  0  0  1  1  0  1  0  1  1  1  1  0
     0  0  0  1  0  0  1  1  0  1  0  1  1  1  1

G =  1  1  0  0  1  0  0  0  0  0  0  0  0  0  0
     0  1  1  0  0  1  0  0  0  0  0  0  0  0  0
     0  0  1  1  0  0  1  0  0  0  0  0  0  0  0
     1  1  0  1  0  0  0  1  0  0  0  0  0  0  0
     1  0  1  0  0  0  0  0  1  0  0  0  0  0  0
     0  1  0  1  0  0  0  0  0  1  0  0  0  0  0
     1  1  1  0  0  0  0  0  0  0  1  0  0  0  0
     0  1  1  1  0  0  0  0  0  0  0  1  0  0  0
     1  1  1  1  0  0  0  0  0  0  0  0  1  0  0
     1  0  1  1  0  0  0  0  0  0  0  0  0  1  0
     1  0  0  1  0  0  0  0  0  0  0  0  0  0  1

n = 15

k = 11
```

Verifica-se que as colunas da matriz **H** são as $15 = 2^4 - 1$ configurações binárias não nulas que se conseguem obter com 4 bits.

As funções MATLAB `encode`, `decode` e `gen2par` também estão relacionadas com a problemática da codificação/descodificação de canal.

6.2 Códigos lineares de bloco cíclicos - uso de polinómios geradores

Relativamente aos códigos cíclicos, estabelecem-se polinómios geradores para códigos (n,k) e obtêm-se as matrizes geradoras e de teste de paridade a partir do polinómio gerador.

O troço de código seguinte mostra o cálculo de polínómios geradores para códigos com dimensões (n,k) especificados como parâmetro na chamada à função `cyclpoly`. Note-se que algumas configurações de valores de n e k, não existem polinómios geradores. Para o caso do código (7,4) existem dois polinómios geradores.

```
>> n=10; k=4; pol = cyclpoly (n, k, 'all')
pol = 1     1     0     0     0     1     1

>> n=9; k=4; pol = cyclpoly (n, k, 'all')
No generator polynomial satisfies the given constraints.
pol = []

>> n=8; k=4; pol = cyclpoly (n, k, 'all')
No generator polynomial satisfies the given constraints.
pol = []

>> n=7; k=4; pol = cyclpoly (n, k, 'all')
pol = 1     0     1     1
```

```
      1     1     0     1

>> n=6; k=4; pol = cyclpoly (n, k, 'all')
pol = 1     1     1

>> n=5; k=4; pol = cyclpoly (n, k, 'all')
pol = 1     1
```

A função `cyclgen` obtém a matriz geradora e de teste de paridade, especificando a dimensão das palavras de código e o polinómio gerador. O vector `pol=[1 1 0 1]` corresponde ao polinómio gerador $g(X) = X^3 + X^2 + 1$.

```
>> n=7;  pol=[1 1 0 1];   [H,G] = cyclgen(n, pol )
H =  1     0     0     1     0     1     1
     0     1     0     1     1     1     0
     0     0     1     0     1     1     1

G =  1     1     0     1     0     0     0
     0     1     1     0     1     0     0
     1     1     1     0     0     1     0
     1     0     1     0     0     0     1
```

Utilizando agora o polinómio gerador $g(X) = X^3 + X + 1$ obtém-se outra matriz geradora e de teste de paridade.

```
>> n=7;  pol=[1 0 1  1];   [H,G] = cyclgen(n, pol )
H =  1     0     0     1     1     1     0
     0     1     0     0     1     1     1
     0     0     1     1     1     0     1

G =  1     0     1     1     0     0     0
     1     1     1     0     1     0     0
     1     1     0     0     0     1     0
     0     1     1     0     0     0     1
```

A divisão polinomial necessária nos códigos cíclicos pode ser realizada através do MATLAB, recorrendo à função `deconv`.

```
>> help deconv
 DECONV Deconvolution and polynomial division.
    [Q,R] = DECONV(B,A) deconvolves vector A out of vector B.  The result
    is returned in vector Q and the remainder in vector R such that
    B = conv(A,Q) + R.

    If A and B are vectors of polynomial coefficients, deconvolution
    is equivalent to polynomial division.  The result of dividing B by
    A is quotient Q and remainder R.
```

A divisão de polinómios $\frac{X^3+1}{X+1}$, apresentada na figura 22, é obtida através do seguinte código. Note-se que é necessário ter em conta que o resultado é definido em aritmética de módulo 2.

```
>> [q,r] = deconv( [1 0 0 1], [1 1])
q = 1    -1     1
r = 0     0     0     0

>> q = mod(q,2)
q = 1     1     1

>> r = mod(r,2)
r = 0     0     0     0
```

A divisão de polinómios $\frac{X^7+1}{X^3+X+1}$, da figura 23 é dada pelo seguinte troço de código.

```
>> [q,r] = deconv( [1 0 0 0 0 0 0 1], [1 0 1 1])
q = 1     0    -1    -1     1
r = 0     0     0     0     0     2     0     0

>> q = mod(q,2)
q = 1     0     1     1     1

>> r = mod(r,2)
r = 0     0     0     0     0     0     0     0
```

Finalmente, a divisão de polinómios $\frac{X^7+1}{X^3+X^2}$, que não tem resto nulo é obtida pela forma que se apresenta de seguida.

```
>> [q,r] = deconv( [1 0 0 0 0 0 0 1], [1 1 0 0])
q = 1    -1     1    -1     1
r = 0     0     0     0     0    -1     0     1

>> mod(q,2)
ans = 1     1     1     1     1

>> mod(r,2)
ans = 0     0     0     0     0     1     0     1
```

7 Aplicações dos códigos

7.1 Códigos lineares de bloco

Nesta secção elencam-se aplicações dos códigos detectores e correctores de erros não cíclicos, referidos neste texto. A tabela 28 apresenta aplicações dos códigos lineares de bloco não cíclicos. Dos exemplos presentes nesta tabela, destaca-se que RAID significa *Redundant Array of Independent Disks* e consiste na utilização de vários discos rígidos para armazenar informação,

garantindo segurança através da utilização dos códigos detectores e correctores de erros. O nível RAID 1 é designado por *mirroring*, o RAID 2 por *Hamming system* e o RAID 3 por *parallel transfer with parity drive*.

Aplicação	**Código utilizado**
Comunicação série assíncrona	1 bit de paridade por cada byte
Memória RAM	1 bit de paridade por cada byte
Memória RAM ECC	Mais do que 1 bit de paridade por cada byte
Teletexto	Hamming (8,4), extensão do Hamming (7,4)
Discos rígidos	Código de Hamming, com bits de paridade por sector
RAID 1	Código de repetição
RAID 2	Com Hamming (7,4) usa 7 discos (4 dados + 3 paridade)
RAID 3	Bit de paridade, vários discos de dados e um de paridade.
Bluetooth	Código de repetição (3,1) para o *packet header* e...
	Hamming modificado (15,10) para a *application data*.

Tabela 28: Aplicações dos códigos lineares de bloco não cíclicos. ECC significa *Error Correcting Code*.

7.2 Códigos lineares de bloco cíclicos

Para os códigos cíclicos, apresentam-se na tabela 29 os polinómios (da tabela 19) e exemplos das respectivas aplicações.

Polinómio	**Aplicação**
CRC32	Codificador de fonte `WinRar`
CRC32	Codificador de fonte `Pkzip` e `WinZip`
CRC32	Protocolo ZMODEM[10]
CRC32	Protocolo Ethernet (norma 802.3)
CRC-CCITT	Protocolo X.25

Tabela 29: Aplicações dos códigos lineares de bloco cíclicos.

Relativamente aos códigos cíclicos importa ainda referir que a família de códigos BCH (Bose-Chaudhuri-Hocqunghem) e os códigos RS (Reed-Solomon) [7, 18] são dos mais utilizados. Os códigos RS são utilizados pelo CD Audio, com a técnica de *interleave* e como tal designa-se por CIRC (*Cross Interleave Reed-Solomon Code*)[11]. O standard de transmissão digital de vídeo DVB (*Digital Video Broadcasting*) também utiliza códigos RS. Em `http://www.eccpage.com` e `http://www.4i2i.com/reed_solomon_codes.htm` encontram-se exemplos de outras aplicações.

[11] `http://www.cdrinfo.com/Sections/Articles/Specific.asp?ArticleHeadline=Writing+Quality&index=0`

A Outros códigos detectores de erros

Apresentam-se nesta secção os algoritmos de cálculo do dígito de controlo do BI (Bilhete de Identidade)BI[12] e do ISBN (International Standard Book Number).

A.1 Dígito do BI

Considerando que o número do BI é representado pelo vector $[m_7\ m_6\ m_5\ m_4\ m_3\ m_2\ m_1\ m_0]$, apresentam-se os algoritmos de cálculo e de verificação do dígito de controlo.

Algoritmo de cálculo

O cálculo do dígito de controlo b_0 é dado por

a. $x = 9m_7 + 8m_6 + 7m_5 + 6m_4 + 5m_3 + 4m_2 + 3m_1 + 2m_0 = \sum_{k=0}^{7}(k+2)m_k$

b. $b_0 = t - x$,

em que t é o primeiro múltiplo de 11 superior a x. Verifica-se assim que $b_0 \in \{0, 1, \ldots, 10\}$.

Algoritmo de verificação

A verificação do dígito de controlo é definida por

a. $s = \sum_{k=0}^{7}(k+2)\hat{m}_k + b_0$

b. caso s seja múltiplo de 11, não se detecta erros; caso contrário, detectam-se erros.

Este código detecta as situações de troca de ordem de dígitos consecutivos, bem como a introdução errada de um dígito, sendo que estes são os erros mais frequentemente introduzidos pelos humanos. Esta capacidade de detecção é conseguida através da diferente ponderação atribuída a cada algarismo.

A.2 Dígito do ISBN

O algoritmo do dígito do BI é semelhante ao do ISBN o qual se apresenta de seguida. O ISBN identifica univocamente livros e publicações não periódicas. Existem duas versões deste número: a mais antiga com 10 dígitos e a mais recente com 13 dígitos. Considerando que o ISBN-10 é representado por $[m_1\ m_2\ \ldots\ m_9\ b_0]$, com 9 dígitos de mensagem e 1 dígito de controlo temos que temos

$$s = 10b_0 + \sum_{k=1}^{9} km_k \tag{53}$$

é divisível por 11.

B Exercícios propostos

1. A tabela de síndromas/padrões de erro, apresentada abaixo, encontra-se incompleta e refere-se a um código de bloco linear sistemático (n,k). O código possui capacidade de correcção de $t = 1$ bit em erro.

[12] `http://www.mat.uc.pt/~picado/SistIdent/mistBI.html`

Síndroma	Padrão de erro
???	0000000
111	1000000
101	0100000
110	???????
011	???????
100	???????
010	???????
???	0000001

a) Complete a tabela e indique as dimensões (n,k).

b) Apresente as matrizes geradora $\mathbf{G}$ e de teste de paridade $\mathbf{H^T}$.

2. Seja o código $(n, 2)$. Estabeleça um código com capacidade de correcção até $t = 1$ bit em erro e indique a dimensão n assim obtida.

3. Considere o código (6,3) com matriz geradora $\mathbf{G} = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$.

a) Pretende-se estender este código adicionando um bit de paridade resultante da soma de todos os bits de mensagem; indique as dimensões (n, k) do código estendido e apresente as matrizes geradora e de teste de paridade.

b) Qual o peso máximo dos padrões de erro que o código estendido consegue: i) detectar? ii) corrigir?

4. Considere a sequência `s=011011110001`, codificada com Hamming (7,4) cuja matriz geradora é

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right].$$

a) Apresente as palavras de código resultantes da codificação da sequência `s`.

b) Indique a distância mínima do código; mostre um exemplo de correcção de 1 bit em erro sobre uma palavra de código.

5. Seja a sub-matriz geradora de paridade P para um código de Hamming (15,11) sistemático determinada de tal forma que o valor numérico de cada linha (interpretado em binário natural) aumenta de cima para baixo.

a) Construa a tabela de síndromas.

b) Mostre que este código é cíclico.

6. Seja o código de bloco linear sistemático (6,4) com as palavras organizadas na forma $c = [m_0\ m_1\ m_2\ m_3\ b_0\ b_1]$. A sub-matriz geradora de paridade é

$$\mathbf{P} = \left[\begin{array}{cc} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{array}\right].$$

a) Apresente as matrizes geradora $\mathbf{G}$ e de teste de paridade $\mathbf{H^T}$. Verifique se as palavras `000000`, `001111` e `011010` pertencem ao código.

b) Calcule a distância mínima (d_{min}) do código e mostre que este não é cíclico.

7. Considere o polinómio gerador $g(X) = X^3 + X + 1$ do código (7,4). Quais das palavras `0000000`, `1011000` e `0000011` pertencem ao código?

8. O polinómio $g(X) = X^3+1$ é gerador dum código de bloco linear (9,3) cíclico e sistemático. Estabeleça o código e a matriz geradora.

9. Quais as vantagens inerentes ao uso de códigos lineares, face aos não lineares? Ilustre essas vantagens em relação ao codificador e descodificador.

10. Quais as vantagens do uso de códigos lineares de bloco cíclicos, face aos códigos lineares de bloco não cíclicos?

11. O polinómio $g(X) = X^4 + X^2 + X + 1$ é gerador do código linear sistemático (7,3).

a) Determine a palavra de código que codifica a mensagem `101`.

b) Qual o síndroma obtido, quando ocorre erro no primeiro bit da palavra de código?

c) Este código tem capacidade de detectar erros de 2 bit e corrigir erros de 1 bit?

12. Considere um sistema RAID de 7 discos, usando código de Hamming (7,4).

a) Mostre que em caso de avaria de um qualquer disco, é possível recuperar toda a informação.

b) Em caso de avaria de dois discos, sabendo quais os discos avariados, mostre que também é possível recuperar toda a informação. *Sugestão: considere as situações em que os dois discos avariados contêm: bits de paridade; bits de mensagem; 1 bit de paridade e 1 bit de mensagem.*

Referências

[1] N. Benvenuto, R. Corvaja, T. Erseghe, and N. Laurenti. *Communication Systems*. Wiley, 2006.

[2] A. Carlson. *Communication Systems*. McGraw-Hill, fourth edition, 2001.

[3] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[4] D. Dummit and R. Foote. *Abstract Algebra*. Prentice Hall, 1991.

[5] F. Halsall. *Data Communications, Computer Networks and Open Systems*. Addison-Wesley, fourth edition, 1995.

[6] R. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 1950.

[7] S. Haykin. *Communication Systems*. John Wiley & Sons, 1994.

[8] L. Magalhães. *Algebra Linear*. Texto Editora, 1997.

[9] D. McKay. *Information Theory, Inference and Learning Algorithms*. Cambridge Press, 2003.

[10] J. Moreira and P. Farrell. *Essentials of Error-Control Coding*. Wiley, 2006.

[11] I. Otung. *Communication Engineering Principles*. Palgrave, 2001.

[12] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, 2nd edition, March 2000.

[13] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July, October 1948.

[14] W. Stallings. *Handbook of Computer Communications Standards*. Stallings-McMillan, 1987.

[15] G. Strang and K. Borre. *Linear Algebra, Geodesy, and GPS*. Wellesley-Cambridge Press, 1997.

[16] Y. Viniotis. *Probability and Random Processes*. McGraw-Hill International Editions, 1997.

[17] D. Welsh. *Codes and Cryptography*. Oxford Science Publications, 1988.

[18] S. Wicker. *Error Control Systems*. Prentice Hall, 1995.